H 5

```
FFFFFFFFFFFFFFF         111               111       XXX         XXX
FFFFFFFFFFFFFFF         111               111       XXX         XXX
FFFFFFFFFFFFFFF         111               111       XXX         XXX
FFF                  111111            111111       XXX         XXX
FFF                  111111            111111       XXX         XXX
FFF                  111111            111111       XXX         XXX
FFF                     111               111          XXX   XXX
FFF                     111               111          XXX   XXX
FFF                     111               111          XXX   XXX
FFFFFFFFFFFF            111               111             XXX
FFFFFFFFFFFF            111               111             XXX
FFFFFFFFFFFF            111               111             XXX
FFF                     111               111          XXX   XXX
FFF                     111               111          XXX   XXX
FFF                     111               111          XXX   XXX
FFF                     111               111       XXX         XXX
FFF                     111               111       XXX         XXX
FFF                     111               111       XXX         XXX
FFF                  111111111         111111111    XXX         XXX
FFF                  111111111         111111111    XXX         XXX
FFF                  111111111         111111111    XXX         XXX
```

```
_$25

Symb
----
IOC1
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_K
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CO
L_CO
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP_
MAP_

MAP
MARK
MARK
MARK
MARK
MARK
```

FILEID**ACLSUBR

```
  AAAAAA      CCCCCCCC  LL           SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR
  AAAAAA      CCCCCCCC  LL           SSSSSSSS  UU      UU  BBBBBBBB    RRRRRRRR
AA      AA  CC          LL         SS          UU      UU  BB      BB  RR      RR
AA      AA  CC          LL         SS          UU      UU  BB      BB  RR      RR
AA      AA  CC          LL         SS          UU      UU  BB      BB  RR      RR
AA      AA  CC          LL         SS          UU      UU  BB      BB  RR      RR
AA      AA  CC          LL           SSSSSS    UU      UU  BBBBBBBB    RRRRRRRR
AA      AA  CC          LL           SSSSSS    UU      UU  BBBBBBBB    RRRRRRRR
AAAAAAAAAA  CC          LL                 SS  UU      UU  BB      BB  RR  RR
AAAAAAAAAA  CC          LL                 SS  UU      UU  BB      BB  RR  RR
AA      AA  CC          LL                 SS  UU      UU  BB      BB  RR    RR      ....
AA      AA  CC          LL                 SS  UU      UU  BB      BB  RR    RR      ....
AA      AA    CCCCCCCC  LLLLLLLLLL  SSSSSSSS    UUUUUUUUUU  BBBBBBBB    RR      RR  ....
AA      AA    CCCCCCCC  LLLLLLLLLL  SSSSSSSS    UUUUUUUUUU  BBBBBBBB    RR      RR  ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

ACLS
V04-

....

```
;     1        0001  0 MODULE ACLSUBR (
;     2        0002  0                 LANGUAGE (BLISS32),
;     3        0003  0                 IDENT = 'V04-000'
;     4        0004  0                 ADDRESSING_MODE (EXTERNAL = GENERAL)
;     5        0005  0                 ) =
;     6        0006  1 BEGIN
;     7        0007  1
;     8        0008  1 !*****************************************************************************
;     9        0009  1 !*                                                                           *
;    10        0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
;    11        0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
;    12        0012  1 !*  ALL RIGHTS RESERVED.                                                      *
;    13        0013  1 !*                                                                           *
;    14        0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
;    15        0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
;    16        0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    17        0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    18        0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    19        0019  1 !*  TRANSFERRED.                                                              *
;    20        0020  1 !*                                                                           *
;    21        0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    22        0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    23        0023  1 !*  CORPORATION.                                                              *
;    24        0024  1 !*                                                                           *
;    25        0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    26        0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                   *
;    27        0027  1 !*                                                                           *
;    28        0028  1 !*                                                                           *
;    29        0029  1 !*****************************************************************************
;    30        0030  1
;    31        0031  1 !++
;    32        0032  1 !
;    33        0033  1 ! FACILITY:     File system subroutines
;    34        0034  1 !
;    35        0035  1 ! ABSTRACT:
;    36        0036  1 !
;    37        0037  1 !       This module contains the subroutines that manage in memory
;    38        0038  1 !       access control lists.
;    39        0039  1 !
;    40        0040  1 ! ENVIRONMENT:
;    41        0041  1 !
;    42        0042  1 !       Modular procedure. No own storage used.
;    43        0043  1 !
;    44        0044  1 !--
;    45        0045  1 !
;    46        0046  1 !
;    47        0047  1 ! AUTHOR:       L. Mark Pilant          CREATION DATE:  30-Sep-1982  11:00
;    48        0048  1 !
;    49        0049  1 ! MODIFIED BY:
;    50        0050  1 !
;    51        0051  1 !       V03-006 LMP0290         L. Mark Pilant,         31-Jul-1984  10:40
;    52        0052  1 !               Make sure ACL_MODENTRY tracks the ACL_LOCATEACE interface
;    53        0053  1 !               change.
;    54        0054  1 !
;    55        0055  1 !       V03-005 LMP0284         L. Mark Pilant,         25-Jul-1984  15:06
;    56        0056  1 !               Add an ACL initialization routine, ACL_INIT_QUEUE.
;    57        0057  1 !
```

```
;    58          0058  1 !       V03-004 LMP0273         L. Mark Pilant,         6-Jul-1984  13:56
;    59          0059  1 !               Fix a bug that caused an ACE to be dropped when the user's
;    60          0060  1 !               buffer filled up during an ACL read.
;    61          0061  1 !
;    62          0062  1 !       V03-003 ACG0426         Andrew C. Goldstein,    4-May-1984  15:14
;    63          0063  1 !               Fix clearing of input buffer in ACL_ERROR call in ACL_ADDENTRY
;    64          0064  1 !
;    65          0065  1 !       V03-002 ACG0418         Andrew C. Goldstein,    19-Apr-1984  13:15
;    66          0066  1 !               Fix returning of NOMOREACE in reading ACL's
;    67          0067  1 !
;    68          0068  1 !       V03-001 ACG0415         Andrew C. Goldstein,    3-Apr-1984  14:33
;    69          0069  1 !               Break out from SYSACLSRV.B32 to make common routines;
;    70          0070  1 !               rework add algorithm to: support multiple ACEs in one
;    71          0071  1 !               add, correctly protect positioning of alarm and audit
;    72          0072  1 !               ACEs at the front of the ACL, fix the block split of
;    73          0073  1 !               large ACLs; general code cleanup and minor bug fixes.
;    74          0074  1 !
;    75          0075  1 !**
;    76          0076  1
;    77          0077  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    78          0078  1 REQUIRE 'SRC$:FCPDEF';
;    79          1069  1
;    80          1070  1
;    81          1071  1 FORWARD ROUTINE
;    82          1072  1         ACL_INIT_QUEUE,                 ! Initialize ACL queue
;    83          1073  1         ACL_ADDENTRY,                   ! add an ACE to an ACL
;    84          1074  1         ACL_DELENTRY,                   ! remove an ACE from an ACL
;    85          1075  1         ACL_MODENTRY,                   ! modify an existing ACE
;    86          1076  1         ACL_FINDENTRY,                  ! locate a specific ACE
;    87          1077  1         ACL_FINDTYPE,                   ! locate a specific type of ACE
;    88          1078  1         ACL_DELETEACL,                  ! remove entire ACL from object
;    89          1079  1         ACL_READACL,                    ! read one or more ACEs
;    90          1080  1         ACL_ACLLENGTH,                  ! determine the size of the ACL
;    91          1081  1         ACL_READACE,                    ! read a single ACE
;    92          1082  1         ACL_LOCATEACE;                  ! locate ACE by context value
;    93          1083  1
;    94          1084  1 EXTERNAL ROUTINE
;    95          1085  1         ALLOC_PAGED,                    ! Paged pool allocator
;    96          1086  1         DALLOC_PAGED;                   ! Paged pool deallocator
;    97          1087  1
;    98          1088  1 MACRO
;    99        M 1089  1         ACL_ERROR (STATUS) =
;   100        M 1090  1                 BEGIN
;   101        M 1091  1                 CH$FILL (0, .COUNT, .ACE);
;   102        M 1092  1                 ACE[ACE$W_FLAGS] = STATUS;
;   103        M 1093  1                 RETURN STATUS;
;   104        M 1094  1                 END
;   105        M 1095  1                 %;
;   106          1096  1
;   107          1097  1 ! Fields used in the ACL context longword.
;   108          1098  1 !
;   109          1099  1 MACRO
;   110          1100  1         CONTEXT_INDEX   = 0, 0, 24, 0 %, ! ACL entry index
;   111          1101  1         CONTEXT_TYPE    = 0, 24, 8, 0 %; ! entry type in use
```

```
;  113          1102  1 %SBTTL  'ACL_INIT_QUEUE - initialize ACL queue head'
;  114          1103  1 GLOBAL ROUTINE ACL_INIT_QUEUE (ORB_ADDRESS) =
;  115          1104  1
;  116          1105  1 !++
;  117          1106  1 !
;  118          1107  1 ! FUNCTIONAL DESCRIPTION:
;  119          1108  1 !
;  120          1109  1 !       This routine is called to initialize an uninitialized ACL queue.
;  121          1110  1 !       If the queue has already been initialized, this routine is a no-op.
;  122          1111  1 !
;  123          1112  1 ! CALLING SEQUENCE:
;  124          1113  1 !       ACL_INIT_QUEUE (ARG1)
;  125          1114  1 !
;  126          1115  1 ! INPUT PARAMETERS:
;  127          1116  1 !       ARG1: address of the ORB
;  128          1117  1 !
;  129          1118  1 ! IMPLICIT INPUTS:
;  130          1119  1 !       none
;  131          1120  1 !
;  132          1121  1 ! OUTPUT PARAMETERS:
;  133          1122  1 !       none
;  134          1123  1 !
;  135          1124  1 ! IMPLICIT OUTPUTS:
;  136          1125  1 !       none
;  137          1126  1 !
;  138          1127  1 ! ROUTINE VALUE:
;  139          1128  1 !       1
;  140          1129  1 !
;  141          1130  1 ! SIDE EFFECTS:
;  142          1131  1 !       ACL queue head is initialized, and the ACL queue bit in the ORB
;  143          1132  1 !       is set.
;  144          1133  1 !
;  145          1134  1 !--
;  146          1135  1
;  147          1136  2 BEGIN
;  148          1137  2
;  149          1138  2 MAP
;  150          1139  2         ORB_ADDRESS     : REF BBLOCK;           ! Address of the ORB
;  151          1140  2
;  152          1141  2 LOCAL
;  153          1142  2         ORB             : REF BBLOCK;           ! Address of the ORB for PRIMARY_FCB
;  154          1143  2
;  155          1144  2 EXTERNAL
;  156          1145  2         CTL$GL_PCB      : REF BBLOCK ADDRESSING_MODE (ABSOLUTE);
;  157          1146  2
;  158          1147  2 LINKAGE
;  159          1148  2         L_MUTEX         = JSB (REGISTER = 0, REGISTER = 4)
;  160          1149  2                         : NOTUSED (5, 6, 7, 8, 9, 10, 11);
;  161          1150  2
;  162          1151  2 EXTERNAL ROUTINE
;  163          1152  2         SCH$LOCKW       : L_MUTEX ADDRESSING_MODE (ABSOLUTE),
;  164          1153  2                                                   ! Lock mutex for write
;  165          1154  2         SCH$UNLOCK      : L_MUTEX ADDRESSING_MODE (ABSOLUTE);
;  166          1155  2                                                   ! Unlock mutex
;  167          1156  2
;  168          1157  2 ! If the ACL queue head is uninitialized, do the initialization now.
;  169          1158  2
```

```
;  170          1159  2 ORB = .ORB_ADDRESS;                          ! For running at elevated IPL
;  171          1160  2 IF NOT .ORB[ORB$V_ACL_QUEUE]
;  172          1161  2 THEN
;  173          1162  3     BEGIN
;  174          1163  3     ORB[ORB$L_ACL_MUTEX] = %X'0000FFFF';       ! Initialize the ACL mutex
;  175          1164  3     SCH$LOCKW (ORB[ORB$L_ACL_MUTEX], .CTL$GL_PCB);
;  176          1165  3     ORB[ORB$V_ACL_QUEUE] = 1;
;  177          1166  3     ORB[ORB$L_ACLFL] = ORB[ORB$L_ACLBL] = ORB[ORB$L_ACLFL];
;  178          1167  3     SCH$UNLOCK (ORB[ORB$L_ACL_MUTEX], .CTL$GL_PCB);
;  179          1168  3     SET_IPL (0);
;  180          1169  2     END;
;  181          1170  2
;  182          1171  2 RETURN 1;
;  183          1172  2
;  184          1173  1 END;                                           ! End of routine ACL_INIT_QUEUE

                                                .TITLE  ACLSUBR
                                                .IDENT  \V04-000\

                                                .EXTRN  ALLOC_PAGED, DALLOC_PAGED
                                                .EXTRN  CTL$GL_PCB, SCH$LOCKW
                                                .EXTRN  SCH$UNLOCK

                                                .PSECT  $CODE$,NOWRT,2

                                    001C 00000  .ENTRY  ACL_INIT_QUEUE, Save R2,R3,R4                  ; 1103
                    53 00000000G    9F   9E 00002  MOVAB   @#CTL$GL_PCB, R3                             ;
                    52          04  AC   D0 00009  MOVL    ORB_ADDRESS, ORB                            ; 1159
              33        0B      A2       01 E0 0000D  BBS     #1, 11(ORB), 1$                         ; 1160
                        04      A2 FFFF  8F 3C 00012  MOVZWL  #65535, 4(ORB)                          ; 1163
                    50          04  A2   9E 00018  MOVAB   4(ORB), R0                                ; 1164
                    54              63   D0 0001C  MOVL    CTL$GL_PCB, R4                              ;
                        00000000G   9F   16 0001F  JSB     @#SCH$LOCKW                                 ;
                        0B      A2       02 88 00025  BISB2   #2, 11(ORB)                            ; 1165
                    50          28  A2   9E 00029  MOVAB   40(ORB), R0                               ; 1166
                        2C      A2       50 D0 0002D  MOVL    R0, 44(ORB)                            ;
                        28      A2       50 D0 00031  MOVL    R0, 40(ORB)                            ;
                    50          04  A2   9E 00035  MOVAB   4(ORB), R0                                ; 1167
                    54              63   D0 00039  MOVL    CTL$GL_PCB, R4                              ;
                        00000000G   9F   16 0003C  JSB     @#SCH$UNLOCK                                ;
                    12              00   DA 00042  MTPR    #0, #18                                     ; 1168
                    50              01   D0 00045 1$:  MOVL    #1, R0                                 ; 1171
                                    04 00048  RET                                                     ; 1173

; Routine Size:  73 bytes,    Routine Base:  $CODE$ + 0000
```

```
;  186     1174  1 %SBTTL  'ACL_ADDENTRY - add an ACE to an ACL'
;  187     1175  1 GLOBAL ROUTINE ACL_ADDENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, LENGTH, ACE_BUFFER) =
;  188     1176  1
;  189     1177  1 !++
;  190     1178  1 !
;  191     1179  1 ! FUNCTIONAL DESCRIPTION:
;  192     1180  1 !
;  193     1181  1 !       This routine is used to add an Access Control Entry to the file ACL.
;  194     1182  1 !       If the ACL context is zero, the ACE is added to the beginning of the
;  195     1183  1 !       ACL. Otherwise, it is inserted into the ACL at the selected place.
;  196     1184  1 !
;  197     1185  1 !       It should be noted that adding an ACE anywhere in the ACL other than
;  198     1186  1 !       the end could possibly result in corruption of the ACL if the system
;  199     1187  1 !       should crash while the new ACE in being inserted.
;  200     1188  1 !
;  201     1189  1 ! CALLING SEQUENCE:
;  202     1190  1 !       ACL_ADDENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, LENGTH, ACE_BUFFER)
;  203     1191  1 !
;  204     1192  1 ! INPUT PARAMETERS:
;  205     1193  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  206     1194  1 !       ACL_CONTEXT: address of ACL context longword
;  207     1195  1 !       LENGTH: size of the user Access Control Entry
;  208     1196  1 !       ACE_BUFFER: address of the user Access Control Entry
;  209     1197  1 !
;  210     1198  1 ! IMPLICIT INPUTS:
;  211     1199  1 !       NONE
;  212     1200  1 !
;  213     1201  1 ! OUTPUT PARAMETERS:
;  214     1202  1 !       NONE
;  215     1203  1 !
;  216     1204  1 ! IMPLICIT OUTPUTS:
;  217     1205  1 !       NONE
;  218     1206  1 !
;  219     1207  1 ! ROUTINE VALUE:
;  220     1208  1 !       1
;  221     1209  1 !
;  222     1210  1 ! SIDE EFFECTS:
;  223     1211  1 !       Access Control Entry inserted in or appended to the file ACL.  If
;  224     1212  1 !       it is an insertion, the ACL context is updated to point after the
;  225     1213  1 !       inserted ACE.
;  226     1214  1 !
;  227     1215  1 !--
;  228     1216  1
;  229     1217  2 BEGIN
;  230     1218  2
;  231     1219  2 MAP
;  232     1220  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  233     1221  2         ACL_CONTEXT     : REF $BBLOCK;          ! Context longword
;  234     1222  2
;  235     1223  2 LABEL
;  236     1224  2         ADD_ENTRY;                              ! Add one ACE to the ACL
;  237     1225  2
;  238     1226  2 LOCAL
;  239     1227  2         COUNT,                                  ! Length of remaining buffer
;  240     1228  2         ACE             : REF $BBLOCK,          ! The address of the user ACE
;  241     1229  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  242     1230  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
```

```
;  243          1231  2          ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  244          1232  2          ACE_NUMBER,                             ! Index of ACE in ACL
;  245          1233  2          ACL_LENGTH,                             ! Length of all ACE's in segment
;  246          1234  2          NEW_ACL         : REF $BBLOCK,          ! Address of the new ACL segment
;  247          1235  2          OLD_CONTEXT     : $BBLOCK [4];          ! Index of existing ACL entry
;  248          1236  2
;  249          1237  2
;  250          1238  2  ! The ACE buffer may contain multiple ACEs. Loop over the ACEs in the buffer,
;  251          1239  2  ! adding them one at a time.
;  252          1240  2
;  253          1241  2  COUNT = .LENGTH;
;  254          1242  2  ACE = .ACE_BUFFER;
;  255          1243  2  UNTIL .COUNT LEQ 0
;  256          1244  2  DO
;  257          1245  3      BEGIN
;  258          1246  4      ADD_ENTRY: BEGIN
;  259          1247  4
;  260          1248  4  ! Sanity check the contents of the ACE - make sure the count field does
;  261          1249  4  ! not exceed the remaining buffer, and that the ACE is at least 4 bytes long.
;  262          1250  4
;  263          1251  4      IF .COUNT LSSU 4
;  264          1252  4      THEN RETURN SS$_BADPARAM;
;  265          1253  4
;  266          1254  4      IF .ACE[ACE$B_SIZE] GTR .COUNT
;  267          1255  4      OR .ACE[ACE$B_SIZE] EQL 0
;  268          1256  4      THEN ACL_ERROR (SS$_IVACL);
;  269          1257  4
;  270          1258  4  ! If the ACE being added is an AUDIT or ALARM ACE, force it to the beginning
;  271          1259  4  ! of the ACL.
;  272          1260  4
;  273          1261  4      ACE_NUMBER = .ACL_CONTEXT[CONTEXT_INDEX];
;  274          1262  4      IF .ACE[ACE$B_TYPE] EQL ACE$C_AUDIT
;  275          1263  4      OR .ACE[ACE$B_TYPE] EQL ACE$C_ALARM
;  276          1264  4      THEN ACE_NUMBER = 0;
;  277          1265  4
;  278          1266  4  ! Determine if the ACE exists already. If it does, the result depends on
;  279          1267  4  ! the relative position of the old and new ACEs. Effectively, we remove
;  280          1268  4  ! the one that is masked by the one preceding it in the ACL.
;  281          1269  4
;  282          1270  4      IF ACL_FINDENTRY (.ACL_QUEUE_HEAD, OLD_CONTEXT, .ACE[ACE$B_SIZE], .ACE, 1)
;  283          1271  4      THEN
;  284          1272  5          BEGIN
;  285          1273  5          IF .OLD_CONTEXT[CONTEXT_INDEX] LSSU .ACE_NUMBER
;  286          1274  5          THEN LEAVE ADD_ENTRY;
;  287          1275  5          ACL_DELENTRY (.ACL_QUEUE_HEAD, OLD_CONTEXT, 0, 0);
;  288          1276  4          END;
;  289          1277  4
;  290          1278  4  ! Now locate the appropriate ACL segment.  If there is no ACL
;  291          1279  4  ! as yet, simply allocate a block of memory and build
;  292          1280  4  ! the new ACL.
;  293          1281  4
;  294          1282  4      IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  295          1283  4      THEN
;  296          1284  5          BEGIN
;  297          1285  5          ACL_POINTER = ALLOC_PAGED (ACL$C_LENGTH + .ACE[ACE$B_SIZE], ACL_TYPE);
;  298          1286  5          CH$MOVE (.ACE[ACE$B_SIZE], .ACE, ACL_POINTER[ACL$L_LIST]);
;  299          1287  5          ACL_POINTER[ACL$W_SIZE] = ACL$C_LENGTH + .ACE[ACE$B_SIZE];
```

```
;  300          1288  5          INSQUE (.ACL_POINTER, .ACL_QUEUE_HEAD[ACL$L_FLINK]);
;  301          1289  5          ACE_NUMBER = 1;
;  302          1290  5          END
;  303          1291  5
;  304          1292  5  ! If there is an existing ACL, position to the location indicated by the
;  305          1293  5  ! context. Then advance over any existing audit or alarm ACEs to ensure
;  306          1294  5  ! that they stay at the front of the ACL. Finally, if we are positioned
;  307          1295  5  ! at the start of a segment, back up to the end of the previous. This prevents
;  308          1296  5  ! successive additions at the same point from fragmenting the ACL.
;  309          1297  5
;  310          1298  4      ELSE
;  311          1299  5          BEGIN
;  312          1300  5          ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACE_NUMBER, ACL_POINTER, ACL_SPLIT);
;  313          1301  5          ACE_POINTER = ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT;
;  314          1302  5          UNTIL ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  315          1303  6             OR (.ACE_POINTER[ACE$B_TYPE] NEQ ACE$C_AUDIT
;  316          1304  6                 AND .ACE_POINTER[ACE$B_TYPE] NEQ ACE$C_ALARM)
;  317          1305  5          DO
;  318          1306  6              BEGIN
;  319          1307  6              ACE_POINTER = .ACE_POINTER + .ACE_POINTER[ACE$B_SIZE];
;  320          1308  6              ACE_NUMBER = .ACE_NUMBER + 1;
;  321          1309  6              IF .ACE_POINTER GEQA .ACL_POINTER + .ACL_POINTER[ACL$W_SIZE]
;  322          1310  6              THEN
;  323          1311  7                  BEGIN
;  324          1312  7                  ACL_POINTER = .ACL_POINTER[ACL$L_FLINK];
;  325          1313  7                  ACE_POINTER = ACL_POINTER[ACL$L_LIST];
;  326          1314  6                  END;
;  327          1315  5              END;
;  328          1316  5
;  329          1317  5          IF .ACE_POINTER EQL ACL_POINTER[ACL$L_LIST]
;  330          1318  5          AND .ACL_POINTER[ACL$L_BLINK] NEQA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  331          1319  5          THEN
;  332          1320  6              BEGIN
;  333          1321  6              ACL_POINTER = .ACL_POINTER[ACL$L_BLINK];
;  334          1322  6              ACE_POINTER = .ACL_POINTER + .ACL_POINTER[ACL$W_SIZE];
;  335          1323  5              END;
;  336          1324  5
;  337          1325  5  ! Now check the size of the segment. If the new entry still fits within
;  338          1326  5  ! the maximum segment size, insert it by allocating a new segment and
;  339          1327  5  ! copying in the pieces.
;  340          1328  5
;  341          1329  5          ACL_SPLIT = .ACE_POINTER - ACL_POINTER[ACL$L_LIST];
;  342          1330  5          ACL_LENGTH = .ACL_POINTER[ACL$W_SIZE] - ACL$C_LENGTH;
;  343          1331  5          IF .ACL_LENGTH + .ACE[ACE$B_SIZE] LEQU MAX_ACL_SIZE
;  344          1332  5          THEN
;  345          1333  6              BEGIN
;  346          1334  6              NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_LENGTH + .ACE[ACE$B_SIZE], ACL_TYPE);
;  347          1335  6              NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_LENGTH + .ACE[ACE$B_SIZE];
;  348          1336  6              ACE_POINTER = CH$MOVE (.ACL_SPLIT, ACL_POINTER[ACL$L_LIST],
;  349          1337  6                                     NEW_ACL[ACL$L_LIST]);
;  350          1338  6              ACE_POINTER = CH$MOVE (.ACE[ACE$B_SIZE], .ACE, .ACE_POINTER);
;  351          1339  6              CH$MOVE (.ACL_LENGTH - .ACL_SPLIT,
;  352          1340  6                       ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT, .ACE_POINTER);
;  353          1341  6              INSQUE (.NEW_ACL, .ACL_POINTER[ACL$L_BLINK]);
;  354          1342  6              END
;  355          1343  6
;  356          1344  6  ! Otherwise we have to split the segment. We put the new ACE in whichever
```

```
;  357     1345  6 ! segment is smaller. Because the max size of an ACE is 256, and the
;  358     1346  6 ! max segment size is 512, we are guaranteed that the new ACE will fit
;  359     1347  6 ! in one or the other (i.e., a 3-way split is not necessary).
;  360     1348  6
;  361     1349  5         ELSE
;  362     1350  6             BEGIN
;  363     1351  6             IF .ACL_SPLIT LEQU .ACL_LENGTH - .ACL_SPLIT
;  364     1352  6             THEN
;  365     1353  7                 BEGIN
;  366     1354  7                 NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_LENGTH - .ACL_SPLIT, ACL_TYPE);
;  367     1355  7                 NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_LENGTH - .ACL_SPLIT;
;  368     1356  7                 CH$MOVE (.ACL_LENGTH - .ACL_SPLIT,
;  369     1357  7                          ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT, NEW_ACL[ACL$L_LIST]);
;  370     1358  7                 INSQUE (.NEW_ACL, ACL_POINTER[ACL$L_FLINK]);
;  371     1359  7                 NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_SPLIT +.ACE[ACE$B_SIZE], ACL_TYPE);
;  372     1360  7                 NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_SPLIT + .ACE[ACE$B_SIZE];
;  373     1361  7                 ACE_POINTER = CH$MOVE (.ACL_SPLIT, ACL_POINTER[ACL$L_LIST],
;  374     1362  7                                       NEW_ACL[ACL$L_LIST]);
;  375     1363  7                 CH$MOVE (.ACE[ACE$B_SIZE], .ACE, .ACE_POINTER);
;  376     1364  7                 INSQUE (.NEW_ACL, ACL_POINTER[ACL$L_FLINK]);
;  377     1365  7                 END
;  378     1366  6             ELSE
;  379     1367  7                 BEGIN
;  380     1368  7                 NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_LENGTH - .ACL_SPLIT +.ACE[ACE$B_SIZE], ACL_TYPE);
;  381     1369  7                 NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_LENGTH - .ACL_SPLIT + .ACE[ACE$B_SIZE];
;  382     1370  7                 ACE_POINTER = CH$MOVE (.ACE[ACE$B_SIZE], .ACE, NEW_ACL[ACL$L_LIST]);
;  383     1371  7                 CH$MOVE (.ACL_LENGTH - .ACL_SPLIT,
;  384     1372  7                          ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT, .ACE_POINTER);
;  385     1373  7                 INSQUE (.NEW_ACL, ACL_POINTER[ACL$L_FLINK]);
;  386     1374  7                 NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_SPLIT, ACL_TYPE);
;  387     1375  7                 NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_SPLIT;
;  388     1376  7                 CH$MOVE (.ACL_SPLIT, ACL_POINTER[ACL$L_LIST], NEW_ACL[ACL$L_LIST]);
;  389     1377  7                 INSQUE (.NEW_ACL, ACL_POINTER[ACL$L_FLINK]);
;  390     1378  6                 END;
;  391     1379  5             END;
;  392     1380  5         REMQUE (.ACL_POINTER, ACL_POINTER);
;  393     1381  5         DALLOC_PAGED (.ACL_POINTER, ACL_TYPE);
;  394     1382  4         END;
;  395     1383  4
;  396     1384  4 ! At this point the ACE has been added to the ACL.  Finish up by setting the
;  397     1385  4 ! ACL context.
;  398     1386  4
;  399     1387  4     IF .ACE[ACE$B_TYPE] EQL ACE$C_AUDIT
;  400     1388  4     OR .ACE[ACE$B_TYPE] EQL ACE$C_ALARM
;  401     1389  4     THEN .ACL_CONTEXT = ..ACL_CONTEXT + 1
;  402     1390  4     ELSE .ACL_CONTEXT = .ACE_NUMBER + 1;
;  403     1391  3     END;                                        ! end of block ADD_ENTRY
;  404     1392  3     COUNT = .COUNT - .ACE[ACE$B_SIZE];
;  405     1393  3     ACE = .ACE + .ACE[ACE$B_SIZE];
;  406     1394  2     END;                                        ! end of ACE processing loop
;  407     1395  2
;  408     1396  2 RETURN 1;
;  409     1397  2
;  410     1398  1 END;                                            ! End of routine ACL_ADDENTRY
```

```
                                                OFFC 00000          .ENTRY  ACL_ADDENTRY, Save R2,R3,R4,R5,R6,R7,R8,R9,-;  1175
                                                                            R10,R11
                                          5E    0C   C2 00002          SUBL2   #12, SP
                                          5B    0C   AC D0 00005          MOVL    LENGTH, COUNT                            1241
                                          58    10   AC D0 00009          MOVL    ACE_BUFFER, ACE                          1242
                                                     5B D5 0000D 1$:       TSTL    COUNT                                    1243
                                                     03 14 0000F           BGTR    2$
                                                  02A2 31 00011           BRW     23$
                                          04         5B D1 00014 2$:       CMPL    COUNT, #4                                1251
                                                     04 1E 00017           BGEQU   3$
                                          50         14 D0 00019           MOVL    #20, R0                                  1252
                                                     04 0001C              RET
      5B          68          08                 00 ED 0001D 3$:       CMPZV   #0, #8, (ACE), COUNT                     1254
                                                     04 14 00022           BGTR    4$
                                                     68 95 00024           TSTB    (ACE)                                    1255
                                                     12 12 00026           BNEQ    5$
      5B          00          6E                 00 2C 00028 4$:       MOVC5   #0, (SP), #0, COUNT, (ACE)               1256
                                                     68    0002D
                        02    A8    21E4         8F B0 0002E           MOVW    #8676, 2(ACE)
                              50    21E4         8F 3C 00034           MOVZWL  #8676, R0
                                                     04 00039              RET
      5A    08    BC          18                 00 EF 0003A 5$:       EXTZV   #0, #24, @ACL_CONTEXT, ACE_NUMBER        1261
                              05          01     A8 91 00040           CMPB    1(ACE), #5                               1262
                                                     06 13 00044           BEQL    6$
                              06          01     A8 91 00046           CMPB    1(ACE), #6                               1263
                                                     02 12 0004A           BNEQ    7$
                                                     5A D4 0004C 6$:       CLRL    ACE_NUMBER                               1264
                                                     01 DD 0004E 7$:       PUSHL   #1                                       1270
                                                     58 DD 00050           PUSHL   ACE
                                          7E         68 9A 00052           MOVZBL  (ACE), -(SP)
                                       0C    AE    9F 00055              PUSHAB  OLD_CONTEXT
                                       04    AC    DD 00058              PUSHL   ACL_QUEUE_HEAD
                                 0000V       CF    05 FB 0005B           CALLS   #5, ACL_FINDENTRY
                                             17    50 E9 00060           BLBC    R0, 9$
      5A          6E                     18       00 ED 00063           CMPZV   #0, #24, OLD_CONTEXT, ACE_NUMBER         1273
                                                     03 1E 00068           BGEQU   8$
                                                  023A 31 0006A           BRW     22$
                                          7E         7C 0006D 8$:          CLRQ    -(SP)                                    1275
                                       08    AE    9F 0006F              PUSHAB  OLD_CONTEXT
                                       04    AC    DD 00072              PUSHL   ACL_QUEUE_HEAD
                                 0000V       CF    04 FB 00075           CALLS   #4, ACL_DELENTRY
                                   04        AC    04 BC D1 0007A 9$:    CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD          1282
                                                     3A 12 0007F           BNEQ    10$
                                                     07 DD 00081           PUSHL   #7                                       1285
                                          7E         68 9A 00083           MOVZBL  (ACE), -(SP)
                                          6E         0C C0 00086           ADDL2   #12, (SP)
                                 00000000G           00 02 FB 00089        CALLS   #2, ALLOC_PAGED
                                       08    AE       50 D0 00090           MOVL    R0, ACL_POINTER
                                             51       68 9A 00094           MOVZBL  (ACE), R1                                1286
                                             50    08 AE D0 00097           MOVL    ACL_POINTER, R0
                                    0C    A0       68 51 28 0009B        MOVC3   R1, (ACE), 12(R0)
                                             50    08 AE D0 000A0           MOVL    ACL_POINTER, R0                          1287
                                       08    A0       68 9B 000A4           MOVZBW  (ACE), 8(R0)
                                       08    A0       0C A0 000A8           ADDW2   #12, 8(R0)
                                             50    04 BC 9E 000AC           MOVAB   @ACL_QUEUE_HEAD, R0                      1288
```

```
                     00    B0    08    BE    0E 000B0          INSQUE  @ACL_POINTER, @0(R0)
                           5A          01    D0 000B5          MOVL    #1, ACE_NUMBER                          ;1289
                                 01D6  31    000B8           BRW     19$                                     ;1282
                                 04    AE    9F 000BB 10$:     PUSHAB  ACL_SPLIT                               ;1300
                                 0C    AE    9F 000BE          PUSHAB  ACL_POINTER
                                       5A    DD 000C1          PUSHL   ACE_NUMBER
                                 04    AC    DD 000C3          PUSHL   ACL_QUEUE_HEAD
                           0000V CF          04    FB 000C6    CALLS   #4, ACL_LOCATEACE
                                 5A          50    D0 000CB    MOVL    R0, ACE_NUMBER
                     53          08    AE    04    AE C1 000CE ADDL3   ACL_SPLIT, ACL_POINTER, R3            ;1301
                                 59    0C    A3    9E 000D4    MOVAB   12(R3), ACE_POINTER
                                 50    08    AE    D0 000D8 11$:     MOVL    ACL_POINTER, R0                 ;1302
                           04    AC          50    D1 000DC    CMPL    R0, ACL_QUEUE_HEAD
                                             2B    13 000E0    BEQL    13$
                                 05          01    A9 91 000E2 CMPB    1(ACE_POINTER), #5                  ;1303
                                             06    13 000E6    BEQL    12$
                                 06          01    A9 91 000E8 CMPB    1(ACE_POINTER), #6                  ;1304
                                             1F    12 000EC    BNEQ    13$
                                 51                69    9A 000EE 12$:     MOVZBL  (ACE_POINTER), R1           ;1307
                                 59                51    C0 000F1    ADDL2   R1, ACE_POINTER
                                       5A          D6    F4          INCL    ACE_NUMBER                      ;1308
                                 51          08    A0    3C 000F6    MOVZWL  8(R0), R1                       ;1309
                                 51                50    C0 000FA    ADDL2   R0, R1
                                 51                59    D1 000FD    CMPL    ACE_POINTER, R1
                                             D6    1F 00100          BLSSU   11$
                           08    AE                60    D0 00102    MOVL    (R0), ACL_POINTER               ;1312
                     59    08    AE                0C    C1 00106    ADDL3   #12, ACL_POINTER, ACE_POINTER   ;1313
                                             CB    11 00108          BRB     11$                             ;1302
                                 50          08    AE    D0 0010D 13$:     MOVL    ACL_POINTER, R0         ;1317
                                 51          0C    A0    9E 00111    MOVAB   12(R0), R1
                                 51                59    D1 00115    CMPL    ACE_POINTER, R1
                                             17    12 00118          BNEQ    14$
                           04    AC          04    A0    D1 0011A    CMPL    4(R0), ACL_QUEUE_HEAD           ;1318
                                             10    13 0011F          BEQL    14$
                           08    AE          04    A0    D0 00121    MOVL    4(R0), ACL_POINTER              ;1321
                                 50          08    AE    D0 00126    MOVL    ACL_POINTER, R0                 ;1322
                                 59          08    A0    3C 0012A    MOVZWL  8(R0), ACE_POINTER
                                 59                50    C0 0012E    ADDL2   R0, ACE_POINTER
                                 50          08    AE    D0 00131 14$:     MOVL    ACL_POINTER, R0         ;1329
                     53                      50    C3 00135    SUBL3   R0, ACE_POINTER, R3
                           04    AE          F4    A3    9E 00139 MOVAB   -12(R3), ACL_SPLIT
                                 56          08    A0    3C 0013E MOVZWL  8(R0), ACL_LENGTH                  ;1330
                                 56                0B    C2 00142 SUBL2   #11, ACL_LENGTH
                                 53                68    9A 00145 MOVZBL  (ACE), R3                          ;1331
                                 52                7643  9E 00148 MOVAB   -(ACL_LENGTH)[R3], R2
                     00000200    8F                52    D1 0014C CMPL    R2, #512
                                                   4E    1A 00153 BGTRU   15$
                                                   07    DD 00155 PUSHL   #7                                 ;1334
                                 0C                A2    9F 00157 PUSHAB  12(R2)
                     00000000G   00                02    FB 0015A CALLS   #2, ALLOC_PAGED
                                 57                50    D0 00161 MOVL    R0, NEW_ACL
                                 50                68    9A 00164 MOVZBL  (ACE), R0                          ;1335
                                 51          0C    A046  9E 00167 MOVAB   12(R0)[ACL_LENGTH], R1
                           08    A7                51    B0 0016C MOVW    R1, 8(NEW_ACL)
                                 50          08    AE    D0 00170 MOVL    ACL_POINTER, R0                   ;1336
               0C    A7          0C    A0    04    AE    28 00174 MOVC3   ACL_SPLIT, 12(R0), 12(NEW_ACL)   ;1337
                                 59                53    D0 00178 MOVL    R3, ACE_POINTER
```

```
                                50              68  9A 0017E         MOVZBL  (ACE), R0                                       ; 1338
                           69   68              50  28 00181         MOVC3   R0, (ACE), (ACE_POINTER)
                                59              53  D0 00185         MOVL    R3, ACE_POINTER
                           52   56          04  AE  C3 00188         SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1339
                           50       08     AE  04  AE  C1 0018D     ADDL3   ACL_SPLIT, ACL_POINTER, R0                      ; 1340
                           69       0C     A0  52  28 00193         MOVC3   R2, 12(R0), (ACE_POINTER)
                                50          08  AE  D0 00198         MOVL    ACL_POINTER, R0                                 ; 1341
                                    04     B0  67  0E 0019C         INSQUE  (NEW_ACL), @4(R0)
                                         00DD  31 001A0             BRW     18$                                             ; 1331
                           52   56          04  AE  C3 001A3 15$:    SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1354
                                52              0C  C0 001A8         ADDL2   #12, R2
                           50   56          04  AE  C3 001AB         SUBL3   ACL_SPLIT, ACL_LENGTH, R0                       ; 1351
                                50          04  AE  D1 001B0         CMPL    ACL_SPLIT, R0
                                              66  1A 001B4         BGTRU   16$
                                              07  DD 001B6         PUSHL   #7                                              ; 1354
                                              52  DD 001B8         PUSHL   R2
                      00000000G 00              02  FB 001BA         CALLS   #2, ALLOC_PAGED
                                57              50  D0 001C1         MOVL    R0, NEW_ACL
                           52   56          04  AE  C3 001C4         SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1355
                      08   A7   52              0C  A1 001C9         ADDW3   #12, R2, 8(NEW_ACL)
                           52   56          04  AE  C3 001CE         SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1356
                           50       08     AE  04  AE  C1 001D3     ADDL3   ACL_SPLIT, ACL_POINTER, R0                      ; 1357
                      0C   A7       0C     A0  52  28 001D9         MOVC3   R2, 12(R0), 12(NEW_ACL)
                                    08     BE  67  0E 001DF         INSQUE  (NEW_ACL), @ACL_POINTER                         ; 1358
                                              07  DD 001E3         PUSHL   #7                                              ; 1359
                                50              68  9A 001E5         MOVZBL  (ACE), R0
                                50          08  AE  C0 001E8         ADDL2   ACL_SPLIT, R0
                                            0C  A0  9F 001EC         PUSHAB  12(R0)
                      00000000G 00              02  FB 001EF         CALLS   #2, ALLOC_PAGED
                                57              50  D0 001F6         MOVL    R0, NEW_ACL
                                50              68  9A 001F9         MOVZBL  (ACE), R0                                       ; 1360
                                50          04  AE  C0 001FC         ADDL2   ACL_SPLIT, R0
                      08   A7   50              0C  A1 00200         ADDW3   #12, R0, 8(NEW_ACL)
                                50          08  AE  D0 00205         MOVL    ACL_POINTER, R0                                 ; 1361
                      0C   A7       0C     A0  04  AE  28 00209     MOVC3   ACL_SPLIT, 12(R0), 12(NEW_ACL)                  ; 1362
                                59              53  D0 00210         MOVL    R3, ACE_POINTER
                                50              68  9A 00213         MOVZBL  (ACE), R0                                       ; 1363
                           69   68              50  28 00216         MOVC3   R0, (ACE), (ACE_POINTER)
                                              60  11 0021A         BRB     17$                                             ; 1364
                                              07  DD 0021C 16$:     PUSHL   #7                                              ; 1368
                                            6342  9F 0021E         PUSHAB  (R3)[R2]
                      00000000G 00              02  FB 00221         CALLS   #2, ALLOC_PAGED
                                57              50  D0 00228         MOVL    R0, NEW_ACL
                           52   56          04  AE  C3 0022B         SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1369
                                50              68  9A 00230         MOVZBL  (ACE), R0
                                52              50  C0 00233         ADDL2   R0, R2
                      08   A7   52              0C  A1 00236         ADDW3   #12, R2, 8(NEW_ACL)
                                50              68  9A 0023B         MOVZBL  (ACE), R0                                       ; 1370
                      0C   A7   68              50  28 0023E         MOVC3   R0, (ACE), 12(NEW_ACL)
                                59              53  D0 00243         MOVL    R3, ACE_POINTER
                           52   56          04  AE  C3 00246         SUBL3   ACL_SPLIT, ACL_LENGTH, R2                       ; 1371
                           50       08     AE  04  AE  C1 0024B     ADDL3   ACL_SPLIT, ACL_POINTER, R0                      ; 1372
                           69       0C     A0  52  28 00251         MOVC3   R2, 12(R0), (ACE_POINTER)
                                    08     BE  67  0E 00256         INSQUE  (NEW_ACL), @ACL_POINTER                         ; 1373
                                              07  DD 0025A         PUSHL   #7                                              ; 1374
                           7E       08     AE  0C  C1 0025C         ADDL3   #12, ACL_SPLIT, -(SP)
                      00000000G 00              02  FB 00261         CALLS   #2, ALLOC_PAGED
```

```
                                   57           50  D0 00268         MOVL    R0, NEW_ACL                          ;
             08   A7         04    AE           0C  A1 0026B         ADDW3   #12, ACL_SPLIT, 8(NEW_ACL)           ; 1375
                                   50     08    AE  D0 00271         MOVL    ACL_POINTER, R0                      ; 1376
             0C   A7         0C    A0     04    AE  28 00275         MOVC3   ACL_SPLIT, 12(R0), 12(NEW_ACL)       ;
                             08    BE           67  0E 0027C 17$:    INSQUE  (NEW_ACL), @ACL_POINTER              ; 1377
                             08    AE     08    BE  0F 00280 18$:    REMQUE  @ACL_POINTER, ACL_POINTER            ; 1380
                                                07  DD 00285         PUSHL   #7                                   ; 1381
                                          0C    AE  DD 00287         PUSHL   ACL_POINTER                          ;
                      00000000G    00           02  FB 0028A         CALLS   #2, DALLOC_PAGED                     ;
                                   05     01    A8  91 00291 19$:    CMPB    1(ACE), #5                           ; 1387
                                                06  13 00295         BEQL    20$                                  ;
                                   06     01    A8  91 00297         CMPB    1(ACE), #6                           ; 1388
                                                05  12 0029B         BNEQ    21$                                  ;
                                          08    BC  D6 0029D 20$:    INCL    @ACL_CONTEXT                         ; 1389
                                                05  11 002A0         BRB     22$                                  ;
                             08    BC     01    AA  9E 002A2 21$:    MOVAB   1(R10), @ACL_CONTEXT                 ; 1390
                                   50           68  9A 002A7 22$:    MOVZBL  (ACE), R0                            ; 1392
                                   5B           50  C2 002AA         SUBL2   R0, COUNT                            ;
                                   50           68  9A 002AD         MOVZBL  (ACE), R0                            ; 1393
                                   58           50  C0 002B0         ADDL2   R0, ACE                              ;
                                              FD57  31 002B3         BRW     1$                                   ; 1243
                                   50           01  D0 002B6 23$:    MOVL    #1, R0                               ; 1396
                                                    04 002B9         RET                                          ; 1398

; Routine Size:  698 bytes,    Routine Base:  $CODE$ + 0049
```

```
;  412          1399  1 %SBTTL  'ACL_DELENTRY - remove an ACE from an ACL'
;  413          1400  1 GLOBAL ROUTINE ACL_DELENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE) =
;  414          1401  1
;  415          1402  1 !++
;  416          1403  1 !
;  417          1404  1 ! FUNCTIONAL DESCRIPTION:
;  418          1405  1 !
;  419          1406  1 !       This routine is used to delete an Access Control Entry from a file ACL.
;  420          1407  1 !       If the ACL context is valid, and no ACE is specified, then the ACE
;  421          1408  1 !       pointed to by the context is removed.  If an ACE is specified,
;  422          1409  1 !       regardless of the ACL context, it is first located and then removed.
;  423          1410  1 !
;  424          1411  1 ! CALLING SEQUENCE:
;  425          1412  1 !       ACL_DELENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE)
;  426          1413  1 !
;  427          1414  1 ! INPUT PARAMETERS:
;  428          1415  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  429          1416  1 !       ACL_CONTEXT: address of ACL context longword
;  430          1417  1 !       COUNT: size of the user Access Control Entry
;  431          1418  1 !       ACE: address of the user Access Control Entry
;  432          1419  1 !
;  433          1420  1 ! IMPLICIT INPUTS:
;  434          1421  1 !       NONE
;  435          1422  1 !
;  436          1423  1 ! OUTPUT PARAMETERS:
;  437          1424  1 !       NONE
;  438          1425  1 !
;  439          1426  1 ! IMPLICIT OUTPUTS:
;  440          1427  1 !       NONE
;  441          1428  1 !
;  442          1429  1 ! ROUTINE VALUE:
;  443          1430  1 !       1
;  444          1431  1 !
;  445          1432  1 ! SIDE EFFECTS:
;  446          1433  1 !       The Specified ACE is removed from the ACL.  If the ACL segment is then
;  447          1434  1 !       empty, it is removed from the chain.  The ACL context is updated to
;  448          1435  1 !       point to the next ACE in the ACL.
;  449          1436  1 !
;  450          1437  1 !--
;  451          1438  1
;  452          1439  2 BEGIN
;  453          1440  2
;  454          1441  2 MAP
;  455          1442  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  456          1443  2         ACL_CONTEXT     : REF $BBLOCK,          ! Context longword
;  457          1444  2         ACE             : REF $BBLOCK;          ! Address of the user ACE
;  458          1445  2
;  459          1446  2 LOCAL
;  460          1447  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  461          1448  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
;  462          1449  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  463          1450  2         ACE_NUMBER,                             ! Index of ACE in ACL
;  464          1451  2         ACL_LENGTH,                             ! Length of all ACE's in segment
;  465          1452  2         NEW_ACL         : REF $BBLOCK;          ! Address of the new ACL segment
;  466          1453  2
;  467          1454  2
;  468          1455  2 ! Sanity check the length of the supplied ACE.
```

```
;  469          1456  2
;  470          1457  2 IF .COUNT LSSU 4
;  471          1458  2 AND .COUNT NEQ 0
;  472          1459  2 THEN RETURN SS$_BADPARAM;
;  473          1460  2
;  474          1461  2 ! Locate the ACE by content if the content is specified.  Note that this
;  475          1462  2 ! will change the context.
;  476          1463  2
;  477          1464  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  478          1465  2 THEN ACL_ERROR (SS$_ACLEMPTY);
;  479          1466  2
;  480          1467  2 IF .COUNT NEQ 0
;  481          1468  2 THEN IF NOT ACL_FINDENTRY (.ACL_QUEUE_HEAD, .ACL_CONTEXT, .COUNT, .ACE, 1)
;  482          1469  2      THEN ACL_ERROR (SS$_NOENTRY);
;  483          1470  2
;  484          1471  2 ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACL_CONTEXT[CONTEXT_INDEX], ACL_POINTER, ACL_SPLIT);
;  485          1472  2 ACE_POINTER = ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT;
;  486          1473  2
;  487          1474  2 ! Having located the ACE, compute the length of the remaining segment.
;  488          1475  2 ! If it is non-null, allocate a new segment and copy in the remaining
;  489          1476  2 ! portions of the old one. Finally deallocate the old segment.
;  490          1477  2
;  491          1478  2 ACL_LENGTH = .ACL_POINTER[ACL$W_SIZE] - ACL$C_LENGTH - .ACE_POINTER[ACE$B_SIZE];
;  492          1479  2 IF .ACL_LENGTH NEQ 0
;  493          1480  2 THEN
;  494          1481  3     BEGIN
;  495          1482  3     NEW_ACL = ALLOC_PAGED (ACL$C_LENGTH + .ACL_LENGTH, ACL_TYPE);
;  496          1483  3     NEW_ACL[ACL$W_SIZE] = ACL$C_LENGTH + .ACL_LENGTH;
;  497          1484  3     CH$MOVE (.ACL_SPLIT, ACL_POINTER[ACL$L_LIST], NEW_ACL[ACL$L_LIST]);
;  498          1485  3     CH$MOVE (.ACL_LENGTH - .ACL_SPLIT,
;  499          1486  3              ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT + .ACE_POINTER[ACE$B_SIZE],
;  500          1487  3              NEW_ACL[ACL$L_LIST] + .ACL_SPLIT);
;  501          1488  3     INSQUE (.NEW_ACL, .ACL_POINTER[ACL$L_BLINK]);
;  502          1489  3     END;
;  503          1490  2
;  504          1491  2 REMQUE (.ACL_POINTER, ACL_POINTER);
;  505          1492  2 DALLOC_PAGED (.ACL_POINTER, ACL_TYPE);
;  506          1493  2
;  507          1494  2 RETURN 1;
;  508          1495  2
;  509          1496  1 END;                                    ! End of routine ACL_DELENTRY


                                        01FC 00000          .ENTRY  ACL_DELENTRY, Save R2,R3,R4,R5,R6,R7,R8     ; 1400
                                    5E  08   C2 00002       SUBL2   #8, SP                                      ;
                                    04  0C   D1 00005       CMPL    COUNT, #4                                   ; 1457
                                             19 00009       BGEQU   1$                                          ;
                                        0C   D5 0000B       TSTL    COUNT                                       ; 1458
                                             13 0000E       BEQL    1$                                          ;
                                    50      D0 00010        MOVL    #20, R0                                     ; 1459
                                             04 00013       RET                                                 ;
                                04  AC  04   D1 00014 1$:    CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD             ; 1464
                                             12 00019       BNEQ    2$                                          ;
    0C  AC          00          6E          2C 0001B        MOVC5   #0, (SP), #0, COUNT, @ACE                   ; 1465
```

```
                              10  BC     00021
                   50         10  AC  D0 00023             MOVL    ACE, R0
           02      A0       09D0  8F  B0 00027             MOVW    #2512, 2(R0)
                   50       09D0  8F  3C 0002D             MOVZWL  #2512, R0
                                      04 00032             RET                                                  :
                              0C  AC  D5 00033  2$:        TSTL    COUNT                                        : 1467
                                  2A  13 00036             BEQL    3$                                           :
                                  01  DD 00038             PUSHL   #1                                           : 1468
                   7E         0C  AC  7D 0003A             MOVQ    COUNT, -(SP)                                 :
                   7E         04  AC  7D 0003E             MOVQ    ACL_QUEUE_HEAD, -(SP)                        :
              0000V  CF           05  FB 00042             CALLS   #5, ACL_FINDENTRY                            :
                   18             50  E8 00047             BLBS    R0, 3$                                       :
  0C   AC       00  6E            00  2C 0004A             MOVC5   #0, (SP), #0, COUNT, @ACE                    : 1469
                              10  BC     00050                                                                  :
                   50         10  AC  D0 00052             MOVL    ACE, R0                                      :
           02      A0       09D8  8F  B0 00056             MOVW    #2520, 2(R0)                                 :
                   50       09D8  8F  3C 0005C             MOVZWL  #2520, R0                                    :
                                      04 00061             RET                                                  :
                                  5E  DD 00062  3$:        PUSHL   SP                                           : 1471
                              08  AE  9F 00064             PUSHAB  ACL_POINTER                                  :
       7E      08  BC         18  00  EF 00067             EXTZV   #0, #24, @ACL_CONTEXT, -(SP)                 :
                              04  AC  DD 0006D             PUSHL   ACL_QUEUE_HEAD                               :
              0000V  CF           04  FB 00070             CALLS   #4, ACL_LOCATEACE                            :
                   56         04  AE  D0 00075             MOVL    ACL_POINTER, R6                              : 1472
                57  56            6E  C1 00079             ADDL3   ACL_SPLIT, R6, R7                            :
                   57             0C  C0 0007D             ADDL2   #12, ACE_POINTER                             :
                   56         08  A6  3C 00080             MOVZWL  8(R6), R6                                    : 1478
                   50             67  9A 00084             MOVZBL  (ACE_POINTER), R0                            :
                   56             50  C2 00087             SUBL2   R0, R6                                       :
                   56             0C  C2 0008A             SUBL2   #12, ACL_LENGTH                              :
                                  3C  13 0008D             BEQL    4$                                           : 1479
                                  07  DD 0008F             PUSHL   #7                                           : 1482
                              0C  A6  9F 00091             PUSHAB  12(ACL_LENGTH)                               :
          00000000G  00           02  FB 00094             CALLS   #2, ALLOC_PAGED                              :
                   58             50  D0 0009B             MOVL    R0, NEW_ACL                                  :
       08   A8     56             0C  A1 0009E             ADDW3   #12, ACL_LENGTH, 8(NEW_ACL)                  : 1483
                   50         04  AE  D0 000A3             MOVL    ACL_POINTER, R0                              : 1484
       0C   A8   0C  A0           6E  28 000A7             MOVC3   ACL_SPLIT, 12(R0), 12(NEW_ACL)               :
                   56             6E  C2 000AD             SUBL2   ACL_SPLIT, R6                                : 1485
            51   04  AE           6E  C1 000B0             ADDL3   ACL_SPLIT, ACL_POINTER, R1                   : 1486
                   52             67  9A 000B5             MOVZBL  (ACE_POINTER), R2                            :
            50     58             6E  C1 000B8             ADDL3   ACL_SPLIT, NEW_ACL, R0                       : 1487
       0C   A0   0C A241          56  28 000BC             MOVC3   R6, 12(R2)[R1], 12(R0)                       :
                   50         04  AE  D0 000C3             MOVL    ACL_POINTER, R0                              : 1488
           04      B0             68  0E 000C7             INSQUE  (NEW_ACL), @4(R0)                            :
           04      AE         04  BE  0F 000CB  4$:        REMQUE  @ACL_POINTER, ACL_POINTER                    : 1491
                                  07  DD 000D0             PUSHL   #7                                           : 1492
                              08  AE  DD 000D2             PUSHL   ACL_POINTER                                  :
          00000000G  00           02  FB 000D5             CALLS   #2, DALLOC_PAGED                             :
                   50             01  D0 000DC             MOVL    #1, R0                                       : 1494
                                      04 000DF             RET                                                  : 1496

; Routine Size:  224 bytes,    Routine Base:  $CODE$ + 0303
```

```
;  511          1497  1 %SBTTL  'ACL_MODENTRY - modify an existing ACE'
;  512          1498  1 GLOBAL ROUTINE ACL_MODENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE) =
;  513          1499  1
;  514          1500  1 !++
;  515          1501  1 !
;  516          1502  1 ! FUNCTIONAL DESCRIPTION:
;  517          1503  1 !
;  518          1504  1 !       This routine is used to replace an Access Control Entry in a file ACL.
;  519          1505  1 !       The entry pointed to by the context is replaced with the ACE given.
;  520          1506  1 !
;  521          1507  1 ! CALLING SEQUENCE:
;  522          1508  1 !       ACL_MODENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE)
;  523          1509  1 !
;  524          1510  1 ! INPUT PARAMETERS:
;  525          1511  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  526          1512  1 !       ACL_CONTEXT: address of ACL context longword
;  527          1513  1 !       COUNT: size of the user Access Control Entry
;  528          1514  1 !       ACE: address of the user Access Control Entry
;  529          1515  1 !
;  530          1516  1 ! IMPLICIT INPUTS:
;  531          1517  1 !       NONE
;  532          1518  1 !
;  533          1519  1 ! OUTPUT PARAMETERS:
;  534          1520  1 !       NONE
;  535          1521  1 !
;  536          1522  1 ! IMPLICIT OUTPUTS:
;  537          1523  1 !       NONE
;  538          1524  1 !
;  539          1525  1 ! ROUTINE VALUE:
;  540          1526  1 !       1
;  541          1527  1 !
;  542          1528  1 ! SIDE EFFECTS:
;  543          1529  1 !       The ACE is replaced with the new one.  This is done by deleting the
;  544          1530  1 !       ACE pointed to by the context and then inserting (adding) the
;  545          1531  1 !       changed ACE.
;  546          1532  1 !
;  547          1533  1 !--
;  548          1534  1
;  549          1535  2 BEGIN
;  550          1536  2
;  551          1537  2 MAP
;  552          1538  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  553          1539  2         ACL_CONTEXT     : REF $BBLOCK,          ! Context longword
;  554          1540  2         ACE             : REF $BBLOCK;          ! Address of user supplied ACE
;  555          1541  2
;  556          1542  2 LOCAL
;  557          1543  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  558          1544  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
;  559          1545  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  560          1546  2         ACE_NUMBER;                             ! Index of ACE in ACL
;  561          1547  2
;  562          1548  2
;  563          1549  2 ! Sanity check the length of the supplied ACE.
;  564          1550  2
;  565          1551  2 IF .COUNT LSSU 4
;  566          1552  2 THEN RETURN SS$_BADPARAM;
;  567          1553  2
```

```
;  568          1554  2 ! Check for something in the ACL.
;  569          1555  2
;  570          1556  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  571          1557  2 THEN ACL_ERROR (SS$_ACLEMPTY);
;  572          1558  2
;  573          1559  2 ! Now locate the ACE to be modified.
;  574          1560  2
;  575          1561  2 ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACL_CONTEXT[CONTEXT_INDEX], ACL_POINTER, ACL_SPLIT);
;  576          1562  2 IF .ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  577          1563  2 AND .ACL_SPLIT EQL .ACL_POINTER[ACL$W_SIZE] - ACL$C_LENGTH
;  578          1564  2 THEN ACL_ERROR (SS$_NOENTRY);
;  579          1565  2
;  580          1566  2 ! Remove the old ACE by context.
;  581          1567  2
;  582          1568  2 ACL_DELENTRY (.ACL_QUEUE_HEAD, .ACL_CONTEXT, 0, 0);
;  583          1569  2
;  584          1570  2 ! Insert the new ACE.
;  585          1571  2
;  586          1572  2 ACL_ADDENTRY (.ACL_QUEUE_HEAD, .ACL_CONTEXT, .COUNT, .ACE);
;  587          1573  2
;  588          1574  2 RETURN 1;
;  589          1575  2
;  590          1576  1 END;                                    ! End of routine ACL_MODENTRY


                                        003C 00000          .ENTRY  ACL_MODENTRY, Save R2,R3,R4,R5          ; 1498
                                    5E  08 C2 00002          SUBL2   #8, SP
                                    04  0C AC D1 00005          CMPL    COUNT, #4                           ; 1551
                                           04 1E 00009          BGEQU   1$
                                    50        14 D0 0000B          MOVL    #20, R0                         ; 1552
                                              04 0000E          RET
                                 04 AC     04 BC D1 0000F 1$:   CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD     ; 1556
                                           18 12 00014          BNEQ    2$
OC  AC        00                   6E        00 2C 00016          MOVC5   #0, (SP), #0, COUNT, @ACE       ; 1557
                                           10 BC    0001C
                                    50     10 AC D0 0001E          MOVL    ACE, R0
                                 02 A0  09D0 8F B0 00022          MOVW    #2512, 2(R0)
                                    50  09D0 8F 3C 00028          MOVZWL  #2512, R0
                                              04 0002D          RET
                                    5E        DD 0002E 2$:      PUSHL   SP                              ; 1561
                                       08     AE 9F 00030          PUSHAB  ACL_POINTER
   7E         08  BC                   18     00 EF 00033          EXTZV   #0, #24, @ACL_CONTEXT, -(SP)
                                       04     AC DD 00039          PUSHL   ACL_QUEUE_HEAD
                              0000V    CF     04 FB 0003C          CALLS   #4, ACL_LOCATEACE               ; R
                                    50     04 AE D0 00041          MOVL    ACL_POINTER, R0                 ; 1562
                                 04 AC        60 D1 00045          CMPL    (R0), ACL_QUEUE_HEAD
                                              24 12 00049          BNEQ    3$
                                    50     08 A0 3C 0004B          MOVZWL  8(R0), R0                       ; 1563
                                    50        0C C2 0004F          SUBL2   #12, R0
                                    50           6E D1 00052          CMPL    ACL_SPLIT, R0
                                              18 12 00055          BNEQ    3$
OC  AC        00                   6E        00 2C 00057          MOVC5   #0, (SP), #0, COUNT, @ACE       ; 1564
                                       10     BC    0005D
                                    50     10 AC D0 0005F          MOVL    ACE, R0
```

```
                          02 A0   09D8   8F B0 00063          MOVW    #2520, 2(R0)                                    ;
                             50   09D8   8F 3C 00069          MOVZWL  #2520, R0                                       ;
                                         04 0006E            RET                                                     ;
                                      7E 7C 0006F 3$:        CLRQ    -(SP)                                           ; 1568
                             7E   04   AC 7D 00071          MOVQ    ACL_QUEUE_HEAD, -(SP)                           ;
                          FEA6 CF      04 FB 00075          CALLS   #4, ACL_DELENTRY                                ;
                             7E   0C   AC 7D 0007A          MOVQ    COUNT, -(SP)                                    ; 1572
                             7E   04   AC 7D 0007E          MOVQ    ACL_QUEUE_HEAD, -(SP)                           ;
                          FBDF CF      04 FB 00082          CALLS   #4, ACL_ADDENTRY                                ;
                             50        01 D0 00087          MOVL    #1, R0                                          ; 1574
                                         04 0008A            RET                                                     ; 1576

; Routine Size:  139 bytes,    Routine Base:  $CODE$ + 03E3
```

```
;  592     1577  1 %SBTTL  'ACL_FINDENTRY - locate a specific ACE'
;  593     1578  1 GLOBAL ROUTINE ACL_FINDENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE, INTERNAL) =
;  594     1579  1
;  595     1580  1 !++
;  596     1581  1 !
;  597     1582  1 ! FUNCTIONAL DESCRIPTION:
;  598     1583  1 !
;  599     1584  1 !       This routine locates the specified Access Control Entry and sets the
;  600     1585  1 !       ACL context accordingly.
;  601     1586  1 !
;  602     1587  1 ! CALLING SEQUENCE:
;  603     1588  1 !       ACL_FINDENTRY (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE, INTERNAL)
;  604     1589  1 !
;  605     1590  1 ! INPUT PARAMETERS:
;  606     1591  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  607     1592  1 !       ACL_CONTEXT: address of ACL context longword
;  608     1593  1 !       COUNT: size of the user Access Control Entry
;  609     1594  1 !       ACE: address of the user Access Control Entry
;  610     1595  1 !       INTERNAL: 0 call generated by a user request
;  611     1596  1 !                 1 call generated within the system
;  612     1597  1 !
;  613     1598  1 ! IMPLICIT INPUTS:
;  614     1599  1 !       NONE
;  615     1600  1 !
;  616     1601  1 ! OUTPUT PARAMETERS:
;  617     1602  1 !       NONE
;  618     1603  1 !
;  619     1604  1 ! IMPLICIT OUTPUTS:
;  620     1605  1 !       NONE
;  621     1606  1 !
;  622     1607  1 ! ROUTINE VALUE:
;  623     1608  1 !       1 if successful
;  624     1609  1 !       0 otherwise
;  625     1610  1 !
;  626     1611  1 ! SIDE EFFECTS:
;  627     1612  1 !       NONE
;  628     1613  1 !
;  629     1614  1 !--
;  630     1615  1
;  631     1616  2 BEGIN
;  632     1617  2
;  633     1618  2 MAP
;  634     1619  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  635     1620  2         ACL_CONTEXT     : REF $BBLOCK,          ! Context longword
;  636     1621  2         ACE             : REF $BBLOCK;          ! Address of user ACE
;  637     1622  2
;  638     1623  2 LOCAL
;  639     1624  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  640     1625  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
;  641     1626  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  642     1627  2         ACE_NUMBER;                             ! Index of ACE in ACL
;  643     1628  2
;  644     1629  2
;  645     1630  2 ! Sanity check the length of the supplied ACE.
;  646     1631  2
;  647     1632  2 IF .COUNT LSSU 4
;  648     1633  2 THEN RETURN SS$_BADPARAM;
```

```
;  649     1634  2
;  650     1635  2 ! Check the length of the supplied ACE to make sure we've been given a
;  651     1636  2 ! complete buffer.
;  652     1637  2
;  653     1638  2 IF .ACE[ACE$B_SIZE] GTRU .COUNT
;  654     1639  2 THEN ACL_ERROR (SS$_IVACL);
;  655     1640  2
;  656     1641  2 ! If there is no ACL present on the file, set the context to zero and return.
;  657     1642  2
;  658     1643  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  659     1644  2 THEN
;  660     1645  3     BEGIN
;  661     1646  3     .ACL_CONTEXT = 0;
;  662     1647  3     IF .INTERNAL THEN RETURN 0 ELSE ACL_ERROR (SS$_ACLEMPTY);
;  663     1648  2     END;
;  664     1649  2
;  665     1650  2 ! Loop through all of the ACL segments trying to locate the specified ACE.
;  666     1651  2
;  667     1652  2 ACE_NUMBER = 0;
;  668     1653  2 ACL_POINTER = ACL_QUEUE_HEAD[ACL$L_FLINK];
;  669     1654  2 DO
;  670     1655  3     BEGIN
;  671     1656  3     ACL_POINTER = .ACL_POINTER[ACL$L_FLINK];
;  672     1657  3     ACE_POINTER = ACL_POINTER[ACL$L_LIST];
;  673     1658  3     UNTIL .ACE_POINTER GEQA .ACL_POINTER + .ACL_POINTER[ACL$W_SIZE]
;  674     1659  3     DO
;  675     1660  4         BEGIN
;  676     1661  4         ACE_NUMBER = .ACE_NUMBER + 1;
;  677     1662  4
;  678     1663  4 ! How we match the ACE is type dependent. Generally speaking, ACEs match
;  679     1664  4 ! on the portion of their content by which they are selected in normal
;  680     1665  4 ! use.
;  681     1666  4
;  682     1667  4         IF
;  683     1668  5             BEGIN
;  684     1669  5             CASE .ACE[ACE$B_TYPE] FROM ACE$C_KEYID TO ACE$C_DIRDEF OF
;  685     1670  5                 SET
;  686     1671  5
;  687     1672  5                 [ACE$C_BIJNL,
;  688     1673  5                  ACE$C_AIJNL,
;  689     1674  5                  ACE$C_ATJNL,
;  690     1675  5                  ACE$C_JNLID,
;  691     1676  5                  ACE$C_DIRDEF]:
;  692     1677  5                         .ACE[ACE$B_TYPE] EQL .ACE_POINTER[ACE$B_TYPE];
;  693     1678  5
;  694     1679  5                 [ACE$C_INFO,
;  695     1680  5                  INRANGE,
;  696     1681  5                  OUTRANGE]:
;  697     1682  5                         CH$EQL (.ACE[ACE$B_SIZE], .ACE,
;  698     1683  5                                 .ACE_POINTER[ACE$B_SIZE], .ACE_POINTER);
;  699     1684  5
;  700     1685  5                 [ACE$C_KEYID]:
;  701     1686  7                         (..ACE AND NOT $FIELDMASK (ACE$V_RESERVED)
;  702     1687  6                          ^ ($BYTEOFFSET (ACE$W_FLAGS)*8))
;  703     1688  5                         EQL
;  704     1689  7                         (..ACE_POINTER AND NOT $FIELDMASK (ACE$V_RESERVED)
;  705     1690  6                          ^ ($BYTEOFFSET (ACE$W_FLAGS)*8))
```

```
;  706          1691  5                           AND CH$EQL (.ACE[ACE$B_SIZE] - $BYTEOFFSET (ACE$L_KEY)
;  707          1692  5                                       - .ACE[ACE$V_RESERVED]*4,
;  708          1693  5                                       ACE[ACE$L_KEY] + .ACE[ACE$V_RESERVED]*4,
;  709          1694  5                                       .ACE_POINTER[ACE$B_SIZE] - $BYTEOFFSET (ACE$L_KEY)
;  710          1695  5                                       - .ACE_POINTER[ACE$V_RESERVED]*4,
;  711          1696  5                                       ACE_POINTER[ACE$L_KEY] + .ACE_POINTER[ACE$V_RESERVED]*4);
;  712          1697  5
;  713          1698  5                     [ACE$C_AUDIT,
;  714          1699  5                      ACE$C_ALARM]:
;  715          1700  5                             ..ACE EQL ..ACE_POINTER
;  716          1701  5                             AND CH$EQL (.ACE[ACE$B_SIZE] - $BYTEOFFSET (ACE$T_AUDITNAME),
;  717          1702  5                                       ACE[ACE$T_AUDITNAME],
;  718          1703  5                                       .ACE[ACE$B_SIZE] - $BYTEOFFSET (ACE$T_AUDITNAME),
;  719          1704  5                                       ACE_POINTER[ACE$T_AUDITNAME]);
;  720          1705  5
;  721          1706  5                         TES
;  722          1707  5                     END
;  723          1708  4                 THEN
;  724          1709  4                     BEGIN
;  725          1710  5                     .ACL_CONTEXT = .ACE_NUMBER;
;  726          1711  5                     ACL_CONTEXT[CONTEXT_TYPE] = .ACE_POINTER[ACE$B_TYPE];
;  727          1712  5                     RETURN 1;
;  728          1713  4                     END;
;  729          1714  4
;  730          1715  4                 ACE_POINTER = .ACE_POINTER + .ACE_POINTER[ACE$B_SIZE];
;  731          1716  3                 END;
;  732          1717  2             END
;  733          1718  2         UNTIL .ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK];
;  734          1719  2         .ACL_CONTEXT = 0;
;  735          1720  2
;  736          1721  2         ! At this point the desired ACE has not been found.  Return failure.
;  737          1722  2
;  738          1723  2         IF .INTERNAL THEN RETURN 0 ELSE ACL_ERROR (SS$_NOENTRY);
;  739          1724  2
;  740          1725  1 END;                                           ! End of routine ACL_FINDENTRY


                                                  01FC 00000          .ENTRY  ACL_FINDENTRY, Save R2,R3,R4,R5,R6,R7,R8    ; 1578
                                        04    0C  AC   D1 00002          CMPL    COUNT, #4                                   ; 1632
                                                  04   1E 00006          BGEQU   1$
                                        50        14   D0 00008          MOVL    #20, R0                                     ; 1633
                                                  04   00008          RET
 0C   AC        10   BC                  08        00   ED 0000C 1$:       CMPZV   #0, #8, @ACE, COUNT                         ; 1638
                                                  18   1B 00013          BLEQU   2$
 0C   AC             00                  6E        00   2C 00015          MOVC5   #0, (SP), #0, COUNT, @ACE                   ; 1639
                                              10  BC   0001B
                                        50    10  AC   D0 0001D          MOVL    ACE, R0
                                  02    A0  21E4  8F   B0 00021          MOVW    #8676, 2(R0)
                                        50  21E4  8F   3C 00027          MOVZWL  #8676, R0
                                                  04   0002C          RET
                                  04    AC    04  BC   D1 0002D 2$:       CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD             ; 1643
                                                  22   12 00032          BNEQ    4$
                                              08  BC   D4 00034          CLRL    @ACL_CONTEXT                                ; 1646
                                        03    14  AC   E9 00037          BLBC    INTERNAL, 3$                                ; 1647
```

```
                                              00EC  31 0003B        BRW     17$                                          ;
 0C   AC         00         6E                  00  2C 0003E 3$:    MOVC5   #0, (SP), #0, COUNT, @ACE                    ;
                                         10     BC     00044                                                             ;
                            50           10     AC  D0 00046        MOVL    ACE, R0                                      ;
                      02    A0         09D0     8F  B0 0004A        MOVW    #2512, 2(R0)                                 ;
                            50         09D0     8F  3C 00050        MOVZWL  #2512, R0                                    ;
                                                04     00055        RET                                                  ;
                                                58  D4 00056 4$:    CLRL    ACE_NUMBER                                   ; 1652
                            55           04     AC  D0 00058        MOVL    ACL_QUEUE_HEAD, ACL_POINTER                  ; 1653
                            55                  65  D0 0005C 5$:    MOVL    (ACL_POINTER), ACL_POINTER                   ; 1656
                            54           0C     A5  9E 0005F        MOVAB   12(R5), ACE_POINTER                          ; 1657
                            50           08     A5  3C 00063 6$:    MOVZWL  8(ACL_POINTER), R0                           ; 1658
                            50                  55  C0 00067        ADDL2   ACL_POINTER, R0                              ;
                            50                  54  D1 0006A        CMPL    ACE_POINTER, R0                              ;
                                                03  1F 0006D        BLSSU   7$                                           ;
                                              00A8  31 0006F        BRW     15$                                          ;
                                                58  D6 00072 7$:    INCL    ACE_NUMBER                                   ; 1661
                            56           10     AC  D0 00074        MOVL    ACE, R6                                      ; 1669
           08               01           01     A6  8F 00078        CASEB   1(R6), #1, #8                                ;
 0020      0020           0020                0027     0007D 8$:    .WORD   11$-8$,-                                     ;
 0020      0012           0072                0072     00085                10$-8$,-                                     ;
                                              0020     0008D                10$-8$,-                                     ;
                                                                            10$-8$,-                                     ;
                                                                            12$-8$,-                                     ;
                                                                            12$-8$,-                                     ;
                                                                            9$-8$,-                                      ;
                                                                            10$-8$,-                                     ;
                                                                            10$-8$                                       ;
                            51                  66  9A 0008F 9$:    MOVZBL  (R6), R1                                     ; 1682
                            50                  64  9A 00092        MOVZBL  (ACE_POINTER), R0                            ; 1683
      50         00         66                  51  2D 00095        CMPC5   R1, (R6), #0, R0, (ACE_POINTER)              ; 1682
                                                64     0009A                                                             ;
                                                63  11 0009B        BRB     13$                                          ;
                      01    A4           01     A6  91 0009D 10$:   CMPB    1(R6), 1(ACE_POINTER)                        ; 1677
                                                5C  11 000A2        BRB     13$                                          ;
                 51         66     000F0000     8F  CB 000A4 11$:   BICL3   #983040, (R6), R1                            ; 1686
                 50         64     000F0000     8F  CB 000AC        BICL3   #983040, (ACE_POINTER), R0                   ; 1689
                            50                  51  D1 000B4        CMPL    R1, R0                                       ;
                                                58  12 000B7        BNEQ    14$                                          ;
                            52                  66  9A 000B9        MOVZBL  (R6), R2                                     ; 1691
      51    02   A6         04                  00  EF 000BC        EXTZV   #0, #4, 2(R6), R1                            ; 1692
                 50         51                  02  78 000C2        ASHL    #2, R1, R0                                   ;
                            52                  50  C2 000C6        SUBL2   R0, R2                                       ;
                            52                  08  C2 000C9        SUBL2   #8, R2                                       ;
                            53                  64  9A 000CC        MOVZBL  (ACE_POINTER), R3                            ; 1694
      50    02   A4         04                  00  EF 000CF        EXTZV   #0, #4, 2(ACE_POINTER), R0                   ; 1695
                 57         50                  02  78 000D5        ASHL    #2, R0, R7                                   ;
                            53                  57  C2 000D9        SUBL2   R7, R3                                       ;
                            53                  08  C2 000DC        SUBL2   #8, R3                                       ;
                                         08   A440  DF 000DF        PUSHAL  8(ACE_POINTER)[R0]                           ; 1691
                                         08   A641  DF 000E3        PUSHAL  8(R6)[R1]                                    ;
      53         00         9E                  52  2D 000E7        CMPC5   R2, @(SP)+, #0, R3, @(SP)+                   ;
                                                9E     000EC                                                             ;
                                                11  11 000ED        BRB     13$                                          ;
                            64                  66  D1 000EF 12$:   CMPL    (R6), (ACE_POINTER)                          ; 1700
                                                1D  12 000F2        BNEQ    14$                                          ;
                            50                  66  9A 000F4        MOVZBL  (R6), R0                                     ; 1701
```

```
                              50          08  C2 000F7          SUBL2   #8, R0                                          :
            08   A4      08   A6          50  29 000FA          CMPC3   R0, 8(R6), 8(ACE_POINTER)                       : 1704
                              0F          12 00100 13$:         BNEQ    14$                                             :
                         08   BC          58  D0 00102          MOVL    ACE_NUMBER, @ACL_CONTEXT                        : 1710
   08   BC          08        18    01    A4  F0 00106          INSV    1(ACE_POINTER), #24, #8, @ACL_CONTEXT           : 1711
                              50          01  D0 0010D          MOVL    #1, R0                                          : 1712
                                          04 00110              RET                                                     :
                              50          64  9A 00111 14$:     MOVZBL  (ACE_POINTER), R0                               : 1715
                              54          50  C0 00114          ADDL2   R0, ACE_POINTER                                 :
                                        FF49  31 00117          BRW     6$                                              : 1658
                         04   AC          65  D1 0011A 15$:     CMPL    (ACL_POINTER), ACL_QUEUE_HEAD                   : 1718
                                          03  13 0011E          BEQL    16$                                             :
                                        FF39  31 00120          BRW     5$                                              :
                                    08    BC  D4 00123 16$:     CLRL    @ACL_CONTEXT                                    : 1719
                              03    14    AC  E9 00126          BLBC    INTERNAL, 18$                                   : 1723
                                          50  D4 0012A 17$:     CLRL    R0                                              :
                                          04 0012C              RET                                                     :
   0C   AC          00        6E          00  2C 0012D 18$:     MOVC5   #0, (SP), #0, COUNT, @ACE                       :
                                    10    BC    00133                                                                   :
                              50    10    AC  D0 00135          MOVL    ACE, R0                                         :
                         02   A0  09D8    8F  B0 00139          MOVW    #2520, 2(R0)                                    :
                              50  09D8    8F  3C 0013F          MOVZWL  #2520, R0                                       :
                                          04 00144              RET                                                     : 1725

; Routine Size:  325 bytes,    Routine Base:  $CODE$ + 046E
```

```
;  742     1726  1 %SBTTL  'ACL_FINDTYPE - locate a specific type of ACE'
;  743     1727  1 GLOBAL ROUTINE ACL_FINDTYPE (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE, INTERNAL) =
;  744     1728  1
;  745     1729  1 !++
;  746     1730  1 !
;  747     1731  1 ! FUNCTIONAL DESCRIPTION:
;  748     1732  1 !
;  749     1733  1 !       This routine locates an Access Control Entry of a specific type.
;  750     1734  1 !       The ACL context is set accordingly.
;  751     1735  1 !
;  752     1736  1 ! CALLING SEQUENCE:
;  753     1737  1 !       ACL_FINDTYPE (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE, INTERNAL)
;  754     1738  1 !
;  755     1739  1 ! INPUT PARAMETERS:
;  756     1740  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  757     1741  1 !       ACL_CONTEXT: address of ACL context longword
;  758     1742  1 !       COUNT: size of the user Access Control Entry
;  759     1743  1 !       ACE: address of the user Access Control Entry
;  760     1744  1 !       INTERNAL: 0 call generated by a user request
;  761     1745  1 !                 1 call generated within the system
;  762     1746  1 !
;  763     1747  1 ! IMPLICIT INPUTS:
;  764     1748  1 !       NONE
;  765     1749  1 !
;  766     1750  1 ! OUTPUT PARAMETERS:
;  767     1751  1 !       NONE
;  768     1752  1 !
;  769     1753  1 ! IMPLICIT OUTPUTS:
;  770     1754  1 !       NONE
;  771     1755  1 !
;  772     1756  1 ! ROUTINE VALUE:
;  773     1757  1 !       1 if successful
;  774     1758  1 !       0 otherwise
;  775     1759  1 !
;  776     1760  1 ! SIDE EFFECTS:
;  777     1761  1 !       NONE
;  778     1762  1 !
;  779     1763  1 !--
;  780     1764  1
;  781     1765  2 BEGIN
;  782     1766  2
;  783     1767  2 MAP
;  784     1768  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  785     1769  2         ACL_CONTEXT     : REF $BBLOCK,          ! Context longword
;  786     1770  2         ACE             : REF $BBLOCK;          ! Address of the user ACE
;  787     1771  2
;  788     1772  2 LOCAL
;  789     1773  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  790     1774  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
;  791     1775  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  792     1776  2         ACE_NUMBER;                             ! Index of ACE in ACL
;  793     1777  2
;  794     1778  2
;  795     1779  2 ! Sanity check the length of the supplied ACE.
;  796     1780  2
;  797     1781  2 IF .COUNT LSSU 4
;  798     1782  2 THEN RETURN SS$_BADPARAM;
```

```
;  799          1783  2
;  800          1784  2 ! Determine if the ACL is empty.  If it is, set the context to zero, indicate
;  801          1785  2 ! a failure by clearing the returning ACE, and return success.
;  802          1786  2
;  803          1787  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  804          1788  2 THEN
;  805          1789  3     BEGIN
;  806          1790  3     .ACL_CONTEXT = 0;
;  807          1791  3     IF .INTERNAL THEN RETURN 0 ELSE ACL_ERROR (SS$_ACLEMPTY);
;  808          1792  2     END;
;  809          1793  2
;  810          1794  2 ! If the search is for an ACE type different from the last ACE type found,
;  811          1795  2 ! start from the beginning of the ACL.  Otherwise, continue the search from
;  812          1796  2 ! the ACE after the last one found.  If the ACE type is found, save the
;  813          1797  2 ! current context and return the contents of the ACE.  If the ACE type was
;  814          1798  2 ! not found, determine whether or not it is the first time through and set
;  815          1799  2 ! the appropriate error status.
;  816          1800  2
;  817          1801  2 IF .ACL_CONTEXT[CONTEXT_TYPE] EQL 0
;  818          1802  2 OR .ACL_CONTEXT[CONTEXT_TYPE] NEQ .ACE[ACE$B_TYPE]
;  819          1803  2 THEN .ACL_CONTEXT = 0;
;  820          1804  2 ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACL_CONTEXT[CONTEXT_INDEX] + 1, ACL_POINTER, ACL_SPLIT);
;  821          1805  2 ACE_POINTER = ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT;
;  822          1806  2
;  823          1807  2 WHILE 1
;  824          1808  2 DO
;  825          1809  3     BEGIN
;  826          1810  3     IF .ACE_POINTER GEQA .ACL_POINTER + .ACL_POINTER[ACL$W_SIZE]
;  827          1811  3     THEN
;  828          1812  4         BEGIN
;  829          1813  4         ACL_POINTER = .ACL_POINTER[ACL$L_FLINK];
;  830          1814  4         ACE_POINTER = ACL_POINTER[ACL$L_LIST];
;  831          1815  3         END;
;  832          1816  3     IF ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  833          1817  3     THEN EXITLOOP;
;  834          1818  3
;  835          1819  3     IF .ACE[ACE$B_TYPE] EQL .ACE_POINTER[ACE$B_TYPE]
;  836          1820  4     AND (IF .ACE[ACE$B_TYPE] NEQ ACE$C_INFO
;  837          1821  4          THEN 1
;  838          1822  4          ELSE .ACE[ACE$V_INFO_TYPE] EQL .ACE_POINTER[ACE$V_INFO_TYPE])
;  839          1823  3     THEN
;  840          1824  4         BEGIN
;  841          1825  4         .ACL_CONTEXT = .ACE_NUMBER;
;  842          1826  4         ACL_CONTEXT[CONTEXT_TYPE] = .ACE_POINTER[ACE$B_TYPE];
;  843          1827  4         CH$COPY (.ACE_POINTER[ACE$B_SIZE], .ACE_POINTER,
;  844          1828  4                  0, .COUNT, .ACE);
;  845          1829  4         RETURN 1;
;  846          1830  3         END;
;  847          1831  3
;  848          1832  3     ACE_POINTER = .ACE_POINTER + .ACE_POINTER[ACE$B_SIZE];
;  849          1833  3     ACE_NUMBER = .ACE_NUMBER + 1;
;  850          1834  2     END;
;  851          1835  2
;  852          1836  2 ! At this point the end of the ACL has been reached. Return failure.
;  853          1837  2
;  854          1838  2 .ACL_CONTEXT = 0;
;  855          1839  2 IF .INTERNAL THEN RETURN 0 ELSE ACL_ERROR (SS$_NOENTRY);
```

```
;  856          1840  2
;  857          1841  1 END;                                        ! End of routine ACL_FINDTYPE


                                                00FC 00000          .ENTRY  ACL_FINDTYPE, Save R2,R3,R4,R5,R6,R7    ; 1727
                                          5E        08 C2 00002     SUBL2   #8, SP
                                          04   0C   AC D1 00005     CMPL    COUNT, #4                               ; 1781
                                                    04 1E 00009     BGEQU   1$
                                          50        14 D0 0000B     MOVL    #20, R0                                 ; 1782
                                                    04 0000E        RET
                                    04    AC    04  BC D1 0000F 1$: CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD         ; 1787
                                                    22 12 00014     BNEQ    3$
                                                08  BC D4 00016     CLRL    @ACL_CONTEXT                            ; 1790
                                          03        14 AC E9 00019     BLBC    INTERNAL, 2$                         ; 1791
                                                 00BA 31 0001D     BRW     11$
   0C   AC          00                    6E        00 2C 00020 2$: MOVC5   #0, (SP), #0, COUNT, @ACE
                                                10  BC    00026
                                          50        10 AC D0 00028     MOVL    ACE, R0
                                    02    A0  09D0  8F B0 0002C     MOVW    #2512, 2(R0)
                                          50  09D0  8F 3C 00032     MOVZWL  #2512, R0
                                                    04 00037        RET
      00        08   BC                  08        18 ED 00038 3$: CMPZV   #24, #8, @ACL_CONTEXT, #0               ; 1801
                                                    10 13 0003E     BEQL    4$
                                          50        10 AC D0 00040     MOVL    ACE, R0                              ; 1802
                                          51        01 A0 9A 00044     MOVZBL  1(R0), R1
      51        08   BC                  08        18 ED 00048     CMPZV   #24, #8, @ACL_CONTEXT, R1
                                                    03 13 0004E     BEQL    5$
                                                08  BC D4 00050 4$: CLRL    @ACL_CONTEXT                            ; 1803
                                          5E        DD 00053 5$: PUSHL   SP                                       ; 1804
                                                08  AE 9F 00055     PUSHAB  ACL_POINTER
7E        08   BC                        18        00 EF 00058     EXTZV   #0, #24, @ACL_CONTEXT, -(SP)
                                                    6E D6 0005E     INCL    (SP)
                                                04  AC DD 00060     PUSHL   ACL_QUEUE_HEAD
                                    0000V   CF      04 FB 00063     CALLS   #4, ACL_LOCATEACE
                                          57        50 D0 00068     MOVL    R0, ACE_NUMBER
                                    56    04  AE    6E C1 00068     ADDL3   ACL_SPLIT, ACL_POINTER, R6              ; 1805
                                          56        0C C0 00070     ADDL2   #12, ACE_POINTER
                                          50  04    AE D0 00073 6$: MOVL    ACL_POINTER, R0                         ; 1810
                                          51  08    A0 3C 00077     MOVZWL  8(R0), R1
                                          51        50 C0 0007B     ADDL2   R0, R1
                                          51        56 D1 0007E     CMPL    ACE_POINTER, R1
                                                    09 1F 00081     BLSSU   7$
                                              04    AE  60 D0 00083     MOVL    (R0), ACL_POINTER                   ; 1813
                                    56        04    AE  0C C1 00087     ADDL3   #12, ACL_POINTER, ACE_POINTER       ; 1814
                                              04    AC  04 AE D1 0008C 7$: CMPL    ACL_POINTER, ACL_QUEUE_HEAD     ; 1816
                                                        40 13 00091     BEQL    10$
                                          50        10 AC D0 00093     MOVL    ACE, R0                              ; 1819
                                              01    A6  01 A0 91 00097     CMPB    1(R0), 1(ACE_POINTER)
                                                        2B 12 0009C     BNEQ    9$
                                                07  01  A0 91 0009E     CMPB    1(R0), #7                           ; 1820
                                                        08 12 000A2     BNEQ    8$
                                    51        02    A6  02 A0 8D 000A4     XORB3   2(R0), 2(ACE_POINTER), R1       ; 1822
                                                    0F  51 93 000AA     BITB    R1, #15
                                                        1A 12 000AD     BNEQ    9$
```

```
                                08  BC      57  D0 000AF 8$:    MOVL    ACE_NUMBER, @ACL_CONTEXT                    ; 1825
     08  BC             08          18      01  A6 F0 000B3     INSV    1(ACE_POINTER), #24, #8, @ACL_CONTEXT       ; 1826
                                     50          66 9A 000BA     MOVZBL  (ACE_POINTER), R0                           ; 1827
     0C  AC             00          66          50 2C 000BD     MOVC5   R0, (ACE_POINTER), #0, COUNT, @ACE          ; 1828
                                         10  BC      000C3
                                     50      01  D0 000C5     MOVL    #1, R0                                      ; 1829
                                             04 000C8         RET
                                     50          66 9A 000C9 9$:     MOVZBL  (ACE_POINTER), R0                       ; 1832
                                     56          50 C0 000CC     ADDL2   R0, ACE_POINTER
                                                 57 D6 000CF     INCL    ACE_NUMBER                                  ; 1833
                                                 A0 11 000D1     BRB     6$                                          ; 1807
                                         08  BC  D4 000D3 10$:    CLRL    @ACL_CONTEXT                                ; 1838
                                     03  14  AC  E9 000D6         BLBC    INTERNAL, 12$                               ; 1839
                                             50  D4 000DA 11$:    CLRL    R0
                                             04 000DC         RET
     0C  AC             00          6E          00 2C 000DD 12$:    MOVC5   #0, (SP), #0, COUNT, @ACE
                                         10  BC      000E3
                                     50  10  AC  D0 000E5         MOVL    ACE, R0
                                02   A0  09D8  8F  B0 000E9         MOVW    #2520, 2(R0)
                                     50  09D8  8F  3C 000EF         MOVZWL  #2520, R0
                                             04 000F4         RET                                                 ; 1841

; Routine Size:  245 bytes,    Routine Base:  $CODE$ + 05B3
```

```
;  859          1842  1 %SBTTL  'ACL_DELETEACL - remove entire ACL from object'
;  860          1843  1 GLOBAL ROUTINE ACL_DELETEACL (ACL_QUEUE_HEAD, ACL_CONTEXT) =
;  861          1844  1
;  862          1845  1 !++
;  863          1846  1 !
;  864          1847  1 ! FUNCTIONAL DESCRIPTION:
;  865          1848  1 !
;  866          1849  1 !       This routine removes all the Access Control Entries from a file,
;  867          1850  1 !       except the ACE that granted the calling user access to the file
;  868          1851  1 !       and any protected ACEs.
;  869          1852  1 !
;  870          1853  1 ! CALLING SEQUENCE:
;  871          1854  1 !       ACL_DELETEACL (ACL_QUEUE_HEAD, ACL_CONTEXT)
;  872          1855  1 !
;  873          1856  1 ! INPUT PARAMETERS:
;  874          1857  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  875          1858  1 !       ACL_CONTEXT: address of ACL context longword
;  876          1859  1 !               Note: A context of zero means an internal call,
;  877          1860  1 !               meaning that protected ACEs are also deleted.
;  878          1861  1 !
;  879          1862  1 ! IMPLICIT INPUTS:
;  880          1863  1 !       NONE
;  881          1864  1 !
;  882          1865  1 ! OUTPUT PARAMETERS:
;  883          1866  1 !       NONE
;  884          1867  1 !
;  885          1868  1 ! IMPLICIT OUTPUTS:
;  886          1869  1 !       NONE
;  887          1870  1 !
;  888          1871  1 ! ROUTINE VALUE:
;  889          1872  1 !       1
;  890          1873  1 !
;  891          1874  1 ! SIDE EFFECTS:
;  892          1875  1 !       All of the ACE's for a file, except for that ACE that granted access
;  893          1876  1 !       to the file and protected ACEs, are removed.  This may or may not be
;  894          1877  1 !       all ACE's depending on whether or not the caller is the file owner.
;  895          1878  1 !       The file header and all extension headers are modified to reflect the
;  896          1879  1 !       new ACL.
;  897          1880  1 !
;  898          1881  1 !--
;  899          1882  1
;  900          1883  2 BEGIN
;  901          1884  2
;  902          1885  2 MAP
;  903          1886  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  904          1887  2         ACL_CONTEXT     : REF $BBLOCK;          ! Context longword
;  905          1888  2
;  906          1889  2 LOCAL
;  907          1890  2         ACL_SEGMENT     : REF $BBLOCK,          ! Address of current segment
;  908          1891  2         NEW_SEGMENT     : REF $BBLOCK,          ! Address of new ACL segment
;  909          1892  2         OLD_SEGMENT     : REF $BBLOCK,          ! Address of old ACL segment
;  910          1893  2         NEW_POINTER     : REF $BBLOCK,          ! Where to put the new ACE
;  911          1894  2         OLD_POINTER     : REF $BBLOCK,          ! Where to get the old ACE
;  912          1895  2         NEW_LENGTH,                             ! Length of new ACL segment
;  913          1896  2         ACE_LENGTH,                             ! Length of protected ACE
;  914          1897  2         DUMMY;                                  ! Throw-away from REMQUE
;  915          1898  2
```

```
;  916          1899  2 ! Loop through removing each ACL segment and deallocate the memory.  At this
;  917          1900  2 ! time, no check is made to see if any ACE within the ACL segment grants
;  918          1901  2 ! access to the file by the caller.
;  919          1902  2
;  920          1903  2 ACL_SEGMENT = .ACL_QUEUE_HEAD[ACL$L_FLINK];
;  921          1904  2 UNTIL .ACL_SEGMENT EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  922          1905  2 DO
;  923          1906  3     BEGIN
;  924          1907  3     OLD_SEGMENT = .ACL_SEGMENT;
;  925          1908  3     ACL_SEGMENT = .ACL_SEGMENT[ACL$L_FLINK];
;  926          1909  3     REMQUE (.OLD_SEGMENT, DUMMY);
;  927          1910  3
;  928          1911  3 ! Preserve the protected ACEs if necessary.
;  929          1912  3
;  930          1913  3     IF .ACL_CONTEXT NEQ 0
;  931          1914  3     THEN
;  932          1915  4         BEGIN
;  933          1916  4         NEW_POINTER = OLD_POINTER = OLD_SEGMENT[ACL$L_LIST];
;  934          1917  4         NEW_LENGTH = 0;
;  935          1918  4         UNTIL .OLD_POINTER GEQA .OLD_SEGMENT + .OLD_SEGMENT[ACL$W_SIZE]
;  936          1919  4         DO
;  937          1920  5             BEGIN
;  938          1921  5             ACE_LENGTH = .OLD_POINTER[ACE$B_SIZE];
;  939          1922  5             IF .OLD_POINTER[ACE$V_PROTECTED]
;  940          1923  5             THEN
;  941          1924  6                 BEGIN
;  942          1925  6                 CH$MOVE (.ACE_LENGTH, .OLD_POINTER, .NEW_POINTER);
;  943          1926  6                 NEW_LENGTH = .NEW_LENGTH + .ACE_LENGTH;
;  944          1927  6                 NEW_POINTER = .NEW_POINTER + .ACE_LENGTH;
;  945          1928  5                 END;
;  946          1929  5             OLD_POINTER = .OLD_POINTER + .ACE_LENGTH;
;  947          1930  4             END;
;  948          1931  4         IF .NEW_LENGTH NEQ 0
;  949          1932  4         THEN
;  950          1933  5             BEGIN
;  951          1934  5             NEW_SEGMENT = ALLOC_PAGED (ACL$C_LENGTH + .NEW_LENGTH, ACL_TYPE);
;  952          1935  5             NEW_SEGMENT[ACL$W_SIZE] = ACL$C_LENGTH + .NEW_LENGTH;
;  953          1936  5             CH$MOVE (.NEW_LENGTH, OLD_SEGMENT[ACL$L_LIST], NEW_SEGMENT[ACL$L_LIST]);
;  954          1937  5             INSQUE (.NEW_SEGMENT, .ACL_SEGMENT[ACL$L_BLINK]);
;  955          1938  4             END;
;  956          1939  3         END;
;  957          1940  3
;  958          1941  3     DALLOC_PAGED (.OLD_SEGMENT, ACL_TYPE);
;  959          1942  2     END;
;  960          1943  2
;  961          1944  2 ! Set the context to zero, and return success.
;  962          1945  2
;  963          1946  2 IF .ACL_CONTEXT NEQ 0
;  964          1947  2 THEN .ACL_CONTEXT = 0;
;  965          1948  2
;  966          1949  2 RETURN 1;
;  967          1950  2
;  968          1951  1 END;                                    ! End of routine ACL_DELETEACL
```

```
                                 0FFC 00000          .ENTRY  ACL_DELETEACL, Save R2,R3,R4,R5,R6,R7,R8,-      ; 1843
                                                             R9,R10,R11
                          5E     08  C2 00002          SUBL2   #8, SP
                          5A 04  BC  D0 00005          MOVL    @ACL_QUEUE_HEAD, ACL_SEGMENT                   ; 1903
                    04    AC     5A  D1 00009 1$:      CMPL    ACL_SEGMENT, ACL_QUEUE_HEAD                    ; 1904
                                 6C  13 0000D          BEQL    6$
                          56     5A  D0 0000F          MOVL    ACL_SEGMENT, OLD_SEGMENT                       ; 1907
                          5A     6A  D0 00012          MOVL    (ACL_SEGMENT), ACL_SEGMENT                     ; 1908
                          6E     66  0F 00015          REMQUE  (OLD_SEGMENT), DUMMY                           ; 1909
                             08  AC  D5 00018          TSTL    ACL_CONTEXT                                    ; 1913
                                 51  13 0001B          BEQL    5$
                          57 0C  A6  9E 0001D          MOVAB   12(R6), OLD_POINTER                            ; 1916
                    04    AE     57  D0 00021          MOVL    OLD_POINTER, NEW_POINTER
                                 59  D4 00025          CLRL    NEW_LENGTH                                     ; 1917
                          50 08  A6  3C 00027 2$:      MOVZWL  8(OLD_SEGMENT), R0                             ; 1918
                          50     56  C0 0002B          ADDL2   OLD_SEGMENT, R0
                          50     57  D1 0002E          CMPL    OLD_POINTER, R0
                                 19  1E 00031          BGEQU   4$
                          5B     67  9A 00033          MOVZBL  (OLD_POINTER), ACE_LENGTH                      ; 1921
                0C        03   A7  01  E1 00036          BBC     #1, 3(OLD_POINTER), 3$                         ; 1922
          04    BE              67  5B  28 0003B          MOVC3   ACE_LENGTH, (OLD_POINTER), @NEW_POINTER        ; 1925
                          59     5B  C0 00040          ADDL2   ACE_LENGTH, NEW_LENGTH                         ; 1926
                    04    AE     5B  C0 00043          ADDL2   ACE_LENGTH, NEW_POINTER                        ; 1927
                          57     5B  C0 00047 3$:      ADDL2   ACE_LENGTH, OLD_POINTER                        ; 1929
                                 DB  11 0004A          BRB     2$                                             ; 1918
                                 59  D5 0004C 4$:      TSTL    NEW_LENGTH                                     ; 1931
                                 1E  13 0004E          BEQL    5$
                                 07  DD 00050          PUSHL   #7                                             ; 1934
                             0C  A9  9F 00052          PUSHAB  12(NEW_LENGTH)
                00000000G    00   02  FB 00055          CALLS   #2, ALLOC_PAGED
                          58     50  D0 0005C          MOVL    R0, NEW_SEGMENT
             08   A8          59  0C  A1 0005F          ADDW3   #12, NEW_LENGTH, 8(NEW_SEGMENT)                ; 1935
             0C   A8     0C   A6  59  28 00064          MOVC3   NEW_LENGTH, 12(OLD_SEGMENT), -                 ; 1936
                                                             12(NEW_SEGMENT)
                    04   BA          68  0E 0006A          INSQUE  (NEW_SEGMENT), @4(ACL_SEGMENT)             ; 1937
                                 07  DD 0006E 5$:      PUSHL   #7                                             ; 1941
                                 56  DD 00070          PUSHL   OLD_SEGMENT
                00000000G    00   02  FB 00072          CALLS   #2, DALLOC_PAGED
                                 8E  11 00079          BRB     1$                                             ; 1904
                             08  AC  D5 0007B 6$:      TSTL    ACL_CONTEXT                                    ; 1946
                                 03  13 0007E          BEQL    7$
                             08  BC  D4 00080          CLRL    @ACL_CONTEXT                                   ; 1947
                          50     01  D0 00083 7$:      MOVL    #1, R0                                         ; 1949
                                 04  00086          RET                                                    ; 1951

; Routine Size:  135 bytes,    Routine Base:  $CODE$ + 06A8
```

```
;  970          1952  1 %SBTTL  'ACL_READACL - read one or more ACEs'
;  971          1953  1 GLOBAL ROUTINE ACL_READACL (ACL_QUEUE_HEAD, ACL_CONTEXT, LENGTH, ACE_BUFFER) =
;  972          1954  1
;  973          1955  1 !++
;  974          1956  1 !
;  975          1957  1 ! FUNCTIONAL DESCRIPTION:
;  976          1958  1 !
;  977          1959  1 !       This routine returns as much of the file ACL as will fit in the user's
;  978          1960  1 !       buffer.  Successive calls will return the unread portions of the ACL
;  979          1961  1 !       until the entire ACL has been read.  If an attempt is made to read
;  980          1962  1 !       beyond the end of the ACL, a error is returned to indicate that there
;  981          1963  1 !       is no more to be read.
;  982          1964  1 !
;  983          1965  1 ! CALLING SEQUENCE:
;  984          1966  1 !       ACL_READACL (ACL_QUEUE_HEAD, ACL_CONTEXT, LENGTH, ACE_BUFFER)
;  985          1967  1 !
;  986          1968  1 ! INPUT PARAMETERS:
;  987          1969  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  988          1970  1 !       ACL_CONTEXT: address of ACL context longword
;  989          1971  1 !       LENGTH: size of the user buffer
;  990          1972  1 !       ACE_BUFFER: address of the user buffer
;  991          1973  1 !
;  992          1974  1 ! IMPLICIT INPUTS:
;  993          1975  1 !       NONE
;  994          1976  1 !
;  995          1977  1 ! OUTPUT PARAMETERS:
;  996          1978  1 !       NONE
;  997          1979  1 !
;  998          1980  1 ! IMPLICIT OUTPUTS:
;  999          1981  1 !       NONE
; 1000          1982  1 !
; 1001          1983  1 ! ROUTINE VALUE:
; 1002          1984  1 !       1 if successful
; 1003          1985  1 !       0 otherwise
; 1004          1986  1 !
; 1005          1987  1 ! SIDE EFFECTS:
; 1006          1988  1 !       The users's buffer is filled with as much of the ACL as will fit.
; 1007          1989  1 !       (Only entire ACE's are transferred.)  The ACL context is then updated
; 1008          1990  1 !       to point to the next available ACE.
; 1009          1991  1 !
; 1010          1992  1 !--
; 1011          1993  1
; 1012          1994  2 BEGIN
; 1013          1995  2
; 1014          1996  2 MAP
; 1015          1997  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
; 1016          1998  2         ACL_CONTEXT     : REF $BBLOCK;          ! Context longword
; 1017          1999  2
; 1018          2000  2 LOCAL
; 1019          2001  2         COUNT,                                  ! Remaining buffer size
; 1020          2002  2         ACE             : REF $BBLOCK,          ! Address of the user ACE buffer
; 1021          2003  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
; 1022          2004  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
; 1023          2005  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
; 1024          2006  2         ACE_NUMBER;                             ! Index of ACE in ACL
; 1025          2007  2
; 1026          2008  2
```

```
;  1027     2009  2 ! Initialize the buffer parameters.
;  1028     2010  2
;  1029     2011  2 COUNT = .LENGTH;
;  1030     2012  2 ACE = .ACE_BUFFER;
;  1031     2013  2
;  1032     2014  2 ! Sanity check the length of the supplied ACE.
;  1033     2015  2
;  1034     2016  2 IF .COUNT LSSU 4
;  1035     2017  2 THEN RETURN SS$_BADPARAM;
;  1036     2018  2
;  1037     2019  2 ! If the ACL is empty, return an error.
;  1038     2020  2
;  1039     2021  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1040     2022  2 THEN
;  1041     2023  3     BEGIN
;  1042     2024  3     .ACL_CONTEXT = 0;
;  1043     2025  3     ACL_ERROR (SS$_ACLEMPTY);
;  1044     2026  3     END;
;  1045     2027  2
;  1046     2028  2 ! Start reading ACE's from next available.  If the ACL context is zero,
;  1047     2029  2 ! start reading ACE's from the beginning of the ACL.  In either case only
;  1048     2030  2 ! fill the user's buffer with as many whole ACE's as will fit.  Then save
;  1049     2031  2 ! the context for the next time through.  An error is given when an attempt
;  1050     2032  2 ! is made to read beyond the end of the ACL.
;  1051     2033  2
;  1052     2034  2 ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACL_CONTEXT[CONTEXT_INDEX] + 1, ACL_POINTER, ACL_SPLIT);
;  1053     2035  2 ACE_POINTER = ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT;
;  1054     2036  2
;  1055     2037  2 WHILE 1
;  1056     2038  2 DO
;  1057     2039  3     BEGIN
;  1058     2040  3     IF .ACE_POINTER GEQA .ACL_POINTER + .ACL_POINTER[ACL$W_SIZE]
;  1059     2041  3     THEN
;  1060     2042  4         BEGIN
;  1061     2043  4         ACL_POINTER = .ACL_POINTER[ACL$L_FLINK];
;  1062     2044  4         ACE_POINTER = ACL_POINTER[ACL$L_LIST];
;  1063     2045  3         END;
;  1064     2046  3     IF ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1065     2047  3     THEN EXITLOOP;
;  1066     2048  3
;  1067     2049  3     IF .ACE_POINTER[ACE$B_SIZE] GTRU .COUNT
;  1068     2050  3     THEN
;  1069     2051  4         BEGIN
;  1070     2052  4         .ACL_CONTEXT = .ACE_NUMBER - 1;
;  1071     2053  4         IF .ACE EQLA .ACE_BUFFER THEN ACL_ERROR (SS$_BUFFEROVF);
;  1072     2054  4         CH$FILL (0, .COUNT, .ACE);
;  1073     2055  4         RETURN 1;
;  1074     2056  3         END;
;  1075     2057  3     CH$MOVE (.ACE_POINTER[ACE$B_SIZE], .ACE_POINTER, .ACE);
;  1076     2058  3     ACE = .ACE + .ACE_POINTER[ACE$B_SIZE];
;  1077     2059  3     COUNT = .COUNT - .ACE_POINTER[ACE$B_SIZE];
;  1078     2060  3
;  1079     2061  3     ACE_POINTER = .ACE_POINTER + .ACE_POINTER[ACE$B_SIZE];
;  1080     2062  3     ACE_NUMBER = .ACE_NUMBER + 1;
;  1081     2063  2     END;
;  1082     2064  2
;  1083     2065  2 ! At this point the end of the ACL has been reached.  Return the ACE's
```

```
;  1084          2066  2 ! gathered so far, and set the context to point of the end in case another
;  1085          2067  2 ! context operation takes place. If nothing was returned (i.e., we were
;  1086          2068  2 ! at the end of the ACL to start with, return the status.
;  1087          2069  2
;  1088          2070  2 .ACL_CONTEXT = .ACE_NUMBER;
;  1089          2071  2 IF .ACE EQLA .ACE_BUFFER
;  1090          2072  2 THEN ACL_ERROR (SS$_NOMOREACE);
;  1091          2073  2
;  1092          2074  2 CH$FILL (0, .COUNT, .ACE);
;  1093          2075  2 RETURN 1;
;  1094          2076  2
;  1095          2077  1 END;                                    ! End of routine ACL_READACL


                                    03FC 00000          .ENTRY  ACL_READACL, Save R2,R3,R4,R5,R6,R7,R8,R9    ; 1953
                              5E    08  C2 00002          SUBL2   #8, SP                                       ;
                              58  0C  AC  D0 00005          MOVL    LENGTH, COUNT                                ; 2011
                              57  10  AC  D0 00009          MOVL    ACE_BUFFER, ACE                              ; 2012
                              04      58  D1 0000D          CMPL    COUNT, #4                                    ; 2016
                                      04  1E 00010          BGEQU   1$                                           ;
                              50      14  D0 00012          MOVL    #20, R0                                      ; 2017
                                      04  00015          RET                                                  ;
                        04    AC    04  BC  D1 00016 1$:      CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD              ; 2021
                                      15  12 0001B          BNEQ    2$                                           ;
                                    08  BC  D4 0001D          CLRL    @ACL_CONTEXT                                 ; 2024
58              00          6E        00  2C 00020          MOVC5   #0, (SP), #0, COUNT, (ACE)                   ; 2025
                                          67  00025                                                           ;
                        02    A7  09D0  8F  B0 00026          MOVW    #2512, 2(ACE)                                ;
                              50  09D0  8F  3C 0002C          MOVZWL  #2512, R0                                    ;
                                      04  00031          RET                                                  ;
                                    5E  DD 00032 2$:      PUSHL   SP                                           ; 2034
                                  08  AE  9F 00034          PUSHAB  ACL_POINTER                                  ;
7E          08  BC              18      00  EF 00037          EXTZV   #0, #24, @ACL_CONTEXT, -(SP)                 ;
                                      6E  D6 0003D          INCL    (SP)                                         ;
                                  04  AC  DD 0003F          PUSHL   ACL_QUEUE_HEAD                               ;
                        0000V     CF      04  FB 00042          CALLS   #4, ACL_LOCATEACE                            ;
                                  59      50  D0 00047          MOVL    R0, ACE_NUMBER                               ;
                        56    04  AE      6E  C1 0004A          ADDL3   ACL_SPLIT, ACL_POINTER, R6                   ; 2035
                                  56      0C  C0 0004F          ADDL2   #12, ACE_POINTER                             ;
                                  50  04  AE  D0 00052 3$:      MOVL    ACL_POINTER, R0                              ; 2040
                                  51  08  A0  3C 00056          MOVZWL  8(R0), R1                                    ;
                                  51      50  C0 0005A          ADDL2   R0, R1                                       ;
                                  51      56  D1 0005D          CMPL    ACE_POINTER, R1                              ;
                                          09  1F 00060          BLSSU   4$                                           ;
                              04  AE      60  D0 00062          MOVL    (R0), ACL_POINTER                            ; 2043
                        56    04  AE      0C  C1 00066          ADDL3   #12, ACL_POINTER, ACE_POINTER                ; 2044
                              04  AC  04  AE  D1 0006B 4$:      CMPL    ACL_POINTER, ACL_QUEUE_HEAD                  ; 2046
                                          41  13 00070          BEQL    6$                                           ;
58              66          08          00  ED 00072          CMPZV   #0, #8, (ACE_POINTER), COUNT                 ; 2049
                                          1D  1B 00077          BLEQU   5$                                           ;
                              08  BC  FF  A9  9E 00079          MOVAB   -1(R9), @ACL_CONTEXT                         ; 2052
                              10  AC      57  D1 0007E          CMPL    ACE, ACE_BUFFER                              ; 2053
                                          4B  12 00082          BNEQ    7$                                           ;
58              00          6E        00  2C 00084          MOVC5   #0, (SP), #0, COUNT, (ACE)                   ;
```

```
                                                67      00089
                   02    A7     0601            8F  B0  0008A          MOVW     #1537, 2(ACE)                              ;
                         50     0601            8F  3C  00090          MOVZWL   #1537, R0                                  ;
                                                04      00095          RET                                                 ;
                         50                     66  9A  00096  5$:     MOVZBL   (ACE_POINTER), R0                          ; 2057
           67            66                     50  28  00099          MOVC3    R0, (ACE_POINTER), (ACE)                   ;
                         50                     66  9A  0009D          MOVZBL   (ACE_POINTER), R0                          ; 2058
                         57                     50  C0  000A0          ADDL2    R0, ACE                                    ;
                         50                     66  9A  000A3          MOVZBL   (ACE_POINTER), R0                          ; 2059
                         58                     50  C2  000A6          SUBL2    R0, COUNT                                  ;
                         50                     66  9A  000A9          MOVZBL   (ACE_POINTER), R0                          ; 2061
                         56                     50  C0  000AC          ADDL2    R0, ACE_POINTER                            ;
                                                59  D6  000AF          INCL     ACE_NUMBER                                 ; 2062
                                                9F  11  000B1          BRB      3$                                         ; 2037
                   08    BC                     59  D0  000B3  6$:     MOVL     ACE_NUMBER, @ACL_CONTEXT                   ; 2070
                   10    AC                     57  D1  000B7          CMPL     ACE, ACE_BUFFER                            ; 2071
                                                12  12  000BB          BNEQ     7$                                         ;
58         00            6E                     00  2C  000BD          MOVC5    #0, (SP), #0, COUNT, (ACE)                 ; 2072
                                                67      000C2
                   02    A7     09E0            8F  B0  000C3          MOVW     #2528, 2(ACE)                              ;
                         50     09E0            8F  3C  000C9          MOVZWL   #2528, R0                                  ;
                                                04      000CE          RET                                                 ;
58         00            6E                     00  2C  000CF  7$:     MOVC5    #0, (SP), #0, COUNT, (ACE)                 ; 2074
                                                67      000D4
                         50                     01  D0  000D5          MOVL     #1, R0                                     ; 2075
                                                04      000D8          RET                                                 ; 2077

; Routine Size:  217 bytes,    Routine Base:  $CODE$ + 072F
```

```
;  1097     2078  1 %SBTTL  'ACL_ACLLENGTH - determine the size of the ACL'
;  1098     2079  1 GLOBAL ROUTINE ACL_ACLLENGTH (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, LENGTH) =
;  1099     2080  1
;  1100     2081  1 !++
;  1101     2082  1 !
;  1102     2083  1 ! FUNCTIONAL DESCRIPTION:
;  1103     2084  1 !
;  1104     2085  1 !       This routine returns the length of the Access Control List for the
;  1105     2086  1 !       specified file.
;  1106     2087  1 !
;  1107     2088  1 ! CALLING SEQUENCE:
;  1108     2089  1 !       ACL_ACLLENGTH (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, LENGTH)
;  1109     2090  1 !
;  1110     2091  1 ! INPUT PARAMETERS:
;  1111     2092  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  1112     2093  1 !       ACL_CONTEXT: address of ACL context longword
;  1113     2094  1 !       COUNT: size of the user Access Control Entry
;  1114     2095  1 !       ACE: address of the user Access Control Entry
;  1115     2096  1 !
;  1116     2097  1 ! IMPLICIT INPUTS:
;  1117     2098  1 !       NONE
;  1118     2099  1 !
;  1119     2100  1 ! OUTPUT PARAMETERS:
;  1120     2101  1 !       NONE
;  1121     2102  1 !
;  1122     2103  1 ! IMPLICIT OUTPUTS:
;  1123     2104  1 !       NONE
;  1124     2105  1 !
;  1125     2106  1 ! ROUTINE VALUE:
;  1126     2107  1 !       1
;  1127     2108  1 !
;  1128     2109  1 ! SIDE EFFECTS:
;  1129     2110  1 !       The length of the ACL is returned.  In addition, the ACL context
;  1130     2111  1 !       is cleared.
;  1131     2112  1 !
;  1132     2113  1 !--
;  1133     2114  1
;  1134     2115  2 BEGIN
;  1135     2116  2
;  1136     2117  2 MAP
;  1137     2118  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  1138     2119  2         ACL_CONTEXT     : REF $BBLOCK;          ! Context longword
;  1139     2120  2
;  1140     2121  2 LOCAL
;  1141     2122  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to the current segment
;  1142     2123  2         ACL_LENGTH;                             ! Length of the ACL
;  1143     2124  2
;  1144     2125  2 ! Calculate the length of the ACL.
;  1145     2126  2
;  1146     2127  2 ACL_LENGTH = 0;
;  1147     2128  2
;  1148     2129  2 ACL_POINTER = .ACL_QUEUE_HEAD[ACL$L_FLINK];
;  1149     2130  2 UNTIL .ACL_POINTER EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1150     2131  2 DO
;  1151     2132  3     BEGIN
;  1152     2133  3     ACL_LENGTH = .ACL_LENGTH + .ACL_POINTER[ACL$W_SIZE] - ACL$C_LENGTH;
;  1153     2134  3     ACL_POINTER = .ACL_POINTER[ACL$L_FLINK];
```

```
;  1154          2135  2     END;
;  1155          2136  2
;  1156          2137  2 ! Return the length of the ACL.
;  1157          2138  2
;  1158          2139  2 CH$COPY (4, ACL_LENGTH, 0, .COUNT, .LENGTH);
;  1159          2140  2 RETURN 1;
;  1160          2141  2
;  1161          2142  1 END;                                    ! End of routine ACL_ACLLENGTH



                                                 003C 00000         .ENTRY  ACL_ACLLENGTH, Save R2,R3,R4,R5          ; 2079
                                               7E  D4 00002         CLRL    ACL_LENGTH                               ; 2127
                                         51    04 BC  D0 00004         MOVL    @ACL_QUEUE_HEAD, ACL_POINTER             ; 2129
                                   04    AC       51  D1 00008 1$:     CMPL    ACL_POINTER, ACL_QUEUE_HEAD              ; 2130
                                                  10  13 0000C         BEQL    2$                                       ;
                                         50    08 A1  3C 0000E         MOVZWL  8(ACL_POINTER), R0                       ; 2133
                                         50       6E  C0 00012         ADDL2   ACL_LENGTH, R0                           ;
                                         6E    F4 A0  9E 00015         MOVAB   -12(R0), ACL_LENGTH                      ;
                                         51       61  D0 00019         MOVL    (ACL_POINTER), ACL_POINTER               ; 2134
                                                  EA  11 0001C         BRB     1$                                       ; 2130
      0C   AC              00                    6E  04  2C 0001E 2$:    MOVC5   #4, ACL_LENGTH, #0, COUNT, @LENGTH       ; 2139
                                                  10 BC       00024    ;
                                         50          01  D0 00026         MOVL    #1, R0                               ; 2140
                                                     04 00029         RET                                              ; 2142

; Routine Size:  42 bytes,    Routine Base:  $CODE$ + 0808
```

```
;  1163     2143  1 %SBTTL  'ACL_READACE - read a single ACE'
;  1164     2144  1 GLOBAL ROUTINE ACL_READACE (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE) =
;  1165     2145  1
;  1166     2146  1 !++
;  1167     2147  1 !
;  1168     2148  1 ! FUNCTIONAL DESCRIPTION:
;  1169     2149  1 !
;  1170     2150  1 !       This routine reads a single ACE at a time from the ACL.  If the
;  1171     2151  1 !       ACE will not fit, the error code SS$_BUFFEROVF is returned as an
;  1172     2152  1 !       ACE error.
;  1173     2153  1 !
;  1174     2154  1 ! CALLING SEQUENCE:
;  1175     2155  1 !       ACL_READACE (ACL_QUEUE_HEAD, ACL_CONTEXT, COUNT, ACE)
;  1176     2156  1 !
;  1177     2157  1 ! INPUT PARAMETERS:
;  1178     2158  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  1179     2159  1 !       ACL_CONTEXT: address of ACL context longword
;  1180     2160  1 !       COUNT: size of the user Access Control Entry
;  1181     2161  1 !       ACE: address of the user Access Control Entry
;  1182     2162  1 !
;  1183     2163  1 ! IMPLICIT INPUTS:
;  1184     2164  1 !       NONE
;  1185     2165  1 !
;  1186     2166  1 ! OUTPUT PARAMETERS:
;  1187     2167  1 !       NONE
;  1188     2168  1 !
;  1189     2169  1 ! IMPLICIT OUTPUTS:
;  1190     2170  1 !       NONE
;  1191     2171  1 !
;  1192     2172  1 ! ROUTINE VALUE:
;  1193     2173  1 !       1 if successful
;  1194     2174  1 !       error code otherwise
;  1195     2175  1 !
;  1196     2176  1 ! SIDE EFFECTS:
;  1197     2177  1 !       The user's buffer is filled with the next ACE if it will fit.
;  1198     2178  1 !       Otherwise an error is indicated.
;  1199     2179  1 !
;  1200     2180  1 !--
;  1201     2181  1
;  1202     2182  2 BEGIN
;  1203     2183  2
;  1204     2184  2 MAP
;  1205     2185  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  1206     2186  2         ACL_CONTEXT     : REF $BBLOCK,          ! Context longword
;  1207     2187  2         ACE             : REF $BBLOCK;          ! Address of user ACE buffer
;  1208     2188  2
;  1209     2189  2 LOCAL
;  1210     2190  2         ACL_POINTER     : REF $BBLOCK,          ! Pointer to current ACL segment
;  1211     2191  2         ACL_SPLIT       : REF $BBLOCK,          ! Offset to current ACE
;  1212     2192  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to current ACE
;  1213     2193  2         ACE_NUMBER;                             ! Index of ACE in ACL
;  1214     2194  2
;  1215     2195  2
;  1216     2196  2 ! Sanity check the length of the supplied ACE.
;  1217     2197  2
;  1218     2198  2 IF .COUNT LSSU 4
;  1219     2199  2 THEN RETURN SS$_BADPARAM;
```

```
;  1220          2200  2
;  1221          2201  2 ! Determine if the ACL is empty.  If it is, set the context to zero, and
;  1222          2202  2 ! indicate a failure by clearing the returning ACE, and return success.
;  1223          2203  2
;  1224          2204  2 IF .ACL_QUEUE_HEAD[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1225          2205  2 THEN
;  1226          2206  3     BEGIN
;  1227          2207  3     .ACL_CONTEXT = 0;
;  1228          2208  3     ACL_ERROR (SS$_ACLEMPTY);
;  1229          2209  3     END;
;  1230          2210  2
;  1231          2211  2 ! Transfer the next available ACE to the user's buffer.  If the user's
;  1232          2212  2 ! buffer is not large enough to contain the ACE, the context is unchanged,
;  1233          2213  2 ! and an error is indicated.
;  1234          2214  2
;  1235          2215  2 ACE_NUMBER = ACL_LOCATEACE (.ACL_QUEUE_HEAD, .ACL_CONTEXT[CONTEXT_INDEX] + 1, ACL_POINTER, ACL_SPLIT);
;  1236          2216  2 IF .ACL_POINTER[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1237          2217  2 AND .ACL_SPLIT EQL .ACL_POINTER[ACL$W_SIZE] - ACL$C_LENGTH
;  1238          2218  2 THEN ACL_ERROR (SS$_NOMOREACE);
;  1239          2219  2 ACE_POINTER = ACL_POINTER[ACL$L_LIST] + .ACL_SPLIT;
;  1240          2220  2
;  1241          2221  2 ! The next available ACE has been located.  Make sure there is room for it.
;  1242          2222  2
;  1243          2223  2 IF .ACE_POINTER[ACE$B_SIZE] GTR .COUNT THEN ACL_ERROR (SS$_BUFFEROVF);
;  1244          2224  2
;  1245          2225  2 ! There is room.  Move it to the user's buffer.
;  1246          2226  2
;  1247          2227  2 CH$COPY (.ACE_POINTER[ACE$B_SIZE], .ACE_POINTER, 0, .COUNT, .ACE);
;  1248          2228  2 .ACL_CONTEXT = .ACE_NUMBER;
;  1249          2229  2
;  1250          2230  2 RETURN 1;
;  1251          2231  2
;  1252          2232  1 END;                                       ! End of routine ACL_READACE


                                        00FC 00000          .ENTRY  ACL_READACE, Save R2,R3,R4,R5,R6,R7     ; 2144
                                  5E    08  C2 00002          SUBL2   #8, SP
                                  04 0C AC  D1 00005          CMPL    COUNT, #4                               ; 2198
                                        04  1E 00009          BGEQU   1$
                                  50        14  D0 0000B          MOVL    #20, R0                             ; 2199
                                            04 0000E          RET
                            04    AC    04  BC  D1 0000F 1$:    CMPL    @ACL_QUEUE_HEAD, ACL_QUEUE_HEAD     ; 2204
                                        1B  12 00014          BNEQ    2$
                                        08  BC  D4 00016          CLRL    @ACL_CONTEXT                        ; 2207
   0C   AC          00                  6E  00  2C 00019          MOVC5   #0, (SP), #0, COUNT, @ACE           ; 2208
                                        10  BC     0001F
                                  50    10  AC  D0 00021          MOVL    ACE, R0
                            02    A0  09D0  8F  B0 00025          MOVW    #2512, 2(R0)
                                  50  09D0  8F  3C 0002B          MOVZWL  #2512, R0
                                            04 00030          RET
                                        5E  DD 00031 2$:    PUSHL   SP                                      ; 2215
                                        08  AE  9F 00033          PUSHAB  ACL_POINTER
      7E        08  BC                  18  00  EF 00036          EXTZV   #0, #24, @ACL_CONTEXT, -(SP)
                                            6E  D6 0003C          INCL    (SP)
```

```
                                                 04    AC   DD 0003E          PUSHL   ACL_QUEUE_HEAD                                    ;
                                          0000V  CF          04   FB 00041          CALLS   #4, ACL_LOCATEACE                          ;
                                                 57          50   D0 00046          MOVL    R0, ACE_NUMBER                             ;
                                                 50    04    AE   D0 00049          MOVL    ACL_POINTER, R0                            ; 2216
                                             04  AC          60   D1 0004D          CMPL    (R0), ACL_QUEUE_HEAD                       ;
                                                             24   12 00051          BNEQ    3$                                         ;
                                                 50    08    A0   3C 00053          MOVZWL  8(R0), R0                                  ; 2217
                                                 50          0C   C2 00057          SUBL2   #12, R0                                    ;
                                                 50          6E   D1 0005A          CMPL    ACL_SPLIT, R0                              ;
                                                             18   12 0005D          BNEQ    3$                                         ;
                 OC  AC          00              6E               00   2C 0005F          MOVC5   #0, (SP), #0, COUNT, @ACE          ; 2218
                                                       10    BC        00065                                                           ;
                                                 50    10    AC   D0 00067          MOVL    ACE, R0                                    ;
                                             02  A0    09E0  8F   B0 0006B          MOVW    #2528, 2(R0)                               ;
                                                 50    09E0  8F   3C 00071          MOVZWL  #2528, R0                                  ;
                                                             04   00076          RET                                                ;
                         56      04  AE                  6E   C1 00077 3$:      ADDL3   ACL_SPLIT, ACL_POINTER, R6                 ; 2219
                                 56          0C  C0 0007C                         ADDL2   #12, ACE_POINTER                           ;
                 OC  AC          66          08  ED 0007F                         CMPZV   #0, #8, (ACE_POINTER), COUNT               ; 2223
                                                 18  15 00085                     BLEQ    4$                                         ;
                 OC  AC          00              6E   00   2C 00087              MOVC5   #0, (SP), #0, COUNT, @ACE                  ;
                                                       10    BC        0008D                                                           ;
                                                 50    10    AC   D0 0008F          MOVL    ACE, R0                                    ;
                                             02  A0    0601  8F   B0 00093          MOVW    #1537, 2(R0)                               ;
                                                 50    0601  8F   3C 00099          MOVZWL  #1537, R0                                  ;
                                                             04   0009E          RET                                                ;
                                                 50              66   9A 0009F 4$:      MOVZBL  (ACE_POINTER), R0                  ; 2227
                 OC  AC          00              66               50   2C 000A2          MOVC5   R0, (ACE_POINTER), #0, COUNT, @ACE ;
                                                       10    BC        000A8                                                           ;
                                             08  BC          57   D0 000AA          MOVL    ACE_NUMBER, @ACL_CONTEXT                   ; 2228
                                                 50          01   D0 000AE          MOVL    #1, R0                                     ; 2230
                                                             04   000B1          RET                                                ; 2232

; Routine Size:  178 bytes,    Routine Base:  $CODE$ + 0832
```

```
;  1254          2233  1 %SBTTL  'ACL_LOCATEACE - locate ACE by context value'
;  1255          2234  1 GLOBAL ROUTINE ACL_LOCATEACE (ACL_QUEUE_HEAD, ACE_INDEX, ACL_POINTER, ACL_SPLIT) =
;  1256          2235  1
;  1257          2236  1 !++
;  1258          2237  1 !
;  1259          2238  1 ! FUNCTIONAL DESCRIPTION:
;  1260          2239  1 !
;  1261          2240  1 !       This routine is used to position to a particular Access Control Entry.
;  1262          2241  1 !       This is done by the context specified. A context of zero results in
;  1263          2242  1 !       positioning to the start of the ACL; a number larger than the ACL
;  1264          2243  1 !       size results in positioning to the end.
;  1265          2244  1 !
;  1266          2245  1 ! CALLING SEQUENCE:
;  1267          2246  1 !       ACL_LOCATEACE (ACL_QUEUE_HEAD, ACE_INDEX, ACL_POINTER, ACL_SPLIT)
;  1268          2247  1 !
;  1269          2248  1 ! INPUT PARAMETERS:
;  1270          2249  1 !       ACL_QUEUE_HEAD: address of queue header for ACL
;  1271          2250  1 !       ACE_INDEX: index number of ACE to locate
;  1272          2251  1 !
;  1273          2252  1 ! IMPLICIT INPUTS:
;  1274          2253  1 !       NONE
;  1275          2254  1 !
;  1276          2255  1 ! OUTPUT PARAMETERS:
;  1277          2256  1 !       ACL_POINTER: address to store pointer to the selected ACL segment
;  1278          2257  1 !       ACL_SPLIT: address to store the offset to the selected ACE
;  1279          2258  1 !
;  1280          2259  1 ! IMPLICIT OUTPUTS:
;  1281          2260  1 !       NONE
;  1282          2261  1 !
;  1283          2262  1 ! ROUTINE VALUE:
;  1284          2263  1 !       0 if the context is invalid (points off the end of the ACL)
;  1285          2264  1 !       the numeric position of the ACE
;  1286          2265  1 !
;  1287          2266  1 ! SIDE EFFECTS:
;  1288          2267  1 !       NONE
;  1289          2268  1 !
;  1290          2269  1 !--
;  1291          2270  1
;  1292          2271  2 BEGIN
;  1293          2272  2
;  1294          2273  2 MAP
;  1295          2274  2         ACL_QUEUE_HEAD  : REF $BBLOCK,          ! Queue header for ACL
;  1296          2275  2         ACL_POINTER     : REF $BBLOCK;          ! Address of the current segment
;  1297          2276  2
;  1298          2277  2 LOCAL
;  1299          2278  2         ACL_SEGMENT     : REF $BBLOCK,          ! Address of the current segment
;  1300          2279  2         ACE_POINTER     : REF $BBLOCK,          ! Pointer to ACE within segment
;  1301          2280  2         ACE_NUMBER;                             ! Position of ACE
;  1302          2281  2
;  1303          2282  2 ! Locate the ACE by context.  IF an append is being done, locate to the
;  1304          2283  2 ! end of the ACL chain.
;  1305          2284  2
;  1306          2285  2 ACE_NUMBER = 0;
;  1307          2286  2 ACL_SEGMENT = ACL_QUEUE_HEAD[ACL$L_FLINK];
;  1308          2287  2 UNTIL .ACL_SEGMENT[ACL$L_FLINK] EQLA ACL_QUEUE_HEAD[ACL$L_FLINK]
;  1309          2288  2 DO
;  1310          2289  3     BEGIN
```

```
;  1311     2290  3      ACL_SEGMENT = .ACL_SEGMENT[ACL$L_FLINK];
;  1312     2291  3      ACE_POINTER = ACL_SEGMENT[ACL$L_LIST];
;  1313     2292  3      UNTIL .ACE_POINTER GEQA .ACL_SEGMENT + .ACL_SEGMENT[ACL$W_SIZE]
;  1314     2293  3      DO
;  1315     2294  4          BEGIN
;  1316     2295  4          ACE_NUMBER = .ACE_NUMBER + 1;
;  1317     2296  4          IF .ACE_INDEX LEQU .ACE_NUMBER
;  1318     2297  4          THEN
;  1319     2298  5              BEGIN
;  1320     2299  5              .ACL_SPLIT = .ACE_POINTER - ACL_SEGMENT[ACL$L_LIST];
;  1321     2300  5              .ACL_POINTER = .ACL_SEGMENT;
;  1322     2301  5              RETURN .ACE_NUMBER;
;  1323     2302  4              END;
;  1324     2303  4          ACE_POINTER = .ACE_POINTER + .ACE_POINTER[ACE$B_SIZE];
;  1325     2304  3          END;
;  1326     2305  2      END;
;  1327     2306  2
;  1328     2307  2  ! The ACE pointed to by the ACL context field does not exist.  Set up to
;  1329     2308  2  ! append the ACE to the end of the ACL.
;  1330     2309  2
;  1331     2310  2  .ACL_SPLIT = .ACL_SEGMENT[ACL$W_SIZE] - ACL$C_LENGTH;
;  1332     2311  2  .ACL_POINTER = .ACL_SEGMENT;
;  1333     2312  2  RETURN .ACE_NUMBER + 1;
;  1334     2313  2
;  1335     2314  1  END;                                   ! End of routine ACL_LOCATEACE


                                 000C 00000          .ENTRY  ACL_LOCATEACE, Save R2,R3                       ; 2234
                             51  D4 00002          CLRL    ACE_NUMBER                                      ; 2285
                     50      04  AC  D0 00004          MOVL    ACL_QUEUE_HEAD, ACL_SEGMENT                     ; 2286
                 04  AC          60  D1 00008 1$:       CMPL    (ACL_SEGMENT), ACL_QUEUE_HEAD                   ; 2287
                                  32  13 0000C          BEQL    4$
                     50          60  D0 0000E          MOVL    (ACL_SEGMENT), ACL_SEGMENT                      ; 2290
                     52      0C  A0  9E 00011          MOVAB   12(R0), ACE_POINTER                             ; 2291
                     53      08  A0  3C 00015 2$:       MOVZWL  8(ACL_SEGMENT), R3                              ; 2292
                     53          50  C0 00019          ADDL2   ACL_SEGMENT, R3
                     53          52  D1 0001C          CMPL    ACE_POINTER, R3
                                 E7  1E 0001F          BGEQU   1$
                             51  D6 00021          INCL    ACE_NUMBER                                      ; 2295
                     51      08  AC  D1 00023          CMPL    ACE_INDEX, ACE_NUMBER                           ; 2296
                                 0F  1A 00027          BGTRU   3$
             53          52      50  C3 00029          SUBL3   ACL_SEGMENT, ACE_POINTER, R3                    ; 2299
         10  BC      F4  A3  9E 0002D          MOVAB   -12(R3), @ACL_SPLIT
         0C  BC          50  D0 00032          MOVL    ACL_SEGMENT, @ACL_POINTER                       ; 2300
                             17  11 00036          BRB     5$                                              ; 2301
                     53          62  9A 00038 3$:       MOVZBL  (ACE_POINTER), R3                               ; 2303
                     52          53  C0 0003B          ADDL2   R3, ACE_POINTER
                                 D5  11 0003E          BRB     2$                                              ; 2292
         10  BC      08  A0  3C 00040 4$:       MOVZWL  8(ACL_SEGMENT), @ACL_SPLIT                      ; 2310
         10  BC          0C  C2 00045          SUBL2   #12, @ACL_SPLIT
         0C  BC          50  D0 00049          MOVL    ACL_SEGMENT, @ACL_POINTER                       ; 2311
                             51  D6 0004D          INCL    R1                                              ; 2312
                     50          51  D0 0004F 5$:       MOVL    R1, R0
                                 04  00052          RET                                                     ; 2314
```

```
; Routine Size:  83 bytes,    Routine Base:  $CODE$ + 08E4

; 1336          2315  1
; 1337          2316  1 END
; 1338          2317  0 ELUDOM



;                               PSECT SUMMARY
;
;       Name                    Bytes                           Attributes
;
;  $CODE$                         2359  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                             Library Statistics
;                                        -------- Symbols --------      Pages       Processing
;       File                             Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1          18619        52         0        1000        00:01.8



;                                       COMMAND QUALIFIERS
;
;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:ACLSUBR/OBJ=OBJ$:ACLSUBR MSRC$:ACLSUBR/UPDATE=(ENH$:ACLSUBR)

; Size:          2359 code + 0 data bytes
; Run Time:         00:43.6
; Elapsed Time:     01:36.9
; Lines/CPU Min:     3192
; Lexemes/CPU-Min: 19767
; Memory Used:  278 pages
; Compilation Complete
```

FCPDEF
B32

ACLCNTRL
LIS

ACLSUBR
LIS

SNDERL
LIS

ACPCNTRL
LIS

F11X

F11BXQP
MAP

ACCESS
LIS

TRUNC
LIS

FILESERV
MAP

WITURN
LIS

SNDSMB
LIS